第3章 信息安全技术

信息安全技术主要涵盖数学、密码学、通信、计算机、人工智能和安全工程等学科知识，是实现计算机网络系统安全通信的技术基础，也是实现通信数据的保密性、完整性、可用性、可控性以及不可否认性的安全技术手段。本章主要内容包括信息安全通信的基础知识、密码学技术基础、安全通信协议基础、常用的加密算法和标准、常用的数字签名算法和标准、密钥管理技术以及身份认证技术。

3.1 信息安全通信概述

信息安全的应用范围比较广，从电子商务交易安全到国家军事、政治以及社会基础设施安全。计算机网络系统与信息安全技术相结合，为电子商务交易提供了一个可靠的安全通信平台，电子交易实体可以通过公开网络传输电子交易数据。信息安全通信的主要目标是实现网络通信数据的保密性、完整性、可用性、可控性以及不可否认性。

3.1.1 信息安全通信的目标

信息安全通信是利用信息安全技术在安全的计算机通信网络上传输数据，保证通信数据的安全性，即通信数据的保密性、完整性、可用性、不可否认性以及可控性等。

- 保密性是指通信内容不被授权以外的人访问和控制，甚至隐蔽信息通信的行为事实。军方对信息保密性的要求远高于商业用户，军方的要求是在现有安全技术水平下尽可能地保证通信系统安全，而商业用户在考虑安全性的同时也会考虑安全系统的成本。
- 完整性是指信息在传输、接收或者储存的过程中未被篡改、未被破坏、未被插入、无延迟、未乱序和无丢失，并且能够正确识别信息是否被篡改以及篡改的具体位置。
- 可用性是指信息资源能够被合法用户随时访问和控制，并且在权限许可范围内对信息资源进行操作和处理。信息资源的可用性在系统遭受网络攻击、病毒感染、系统崩溃、战争和自然破坏时显得尤为重要。
- 不可否认性是指信息的处理者不能否认其曾经对信息做过的处理行为，主要目的是防止通信实体日后否认网络通信行为。不可否认性能够防止抵赖，是电子交易系统必须具备的功能，例如在电子商务交易过程中形成的电子合同必须具备不可否认性。
- 可控性是指能够对通信数据和通信系统实施可靠的控制，主要包括授权、审计、责任认定、传播源追踪和监管等控制。可控性是能够限制和控制主体对系统或数据的访问，避免非授权用户的登录、阅读、修改、执行等非法操作。

3.1.2　信息安全通信的基本原理

信息安全通信的参与实体可以分为发送方和接收方。发送方通过数字签名和加密技术将明文信息转换成密文在公开的网络上传输，接收方收到发送方发送的密文信息，首先对密文信息进行解密处理，然后再对接收到的信息进行身份认证和完整性校验，验证无误后接收方接受发送方的数据。如图 3-1 所示的网络通信协议是利用数字证书实现网络安全通信的具体过程，该通信协议需要通信双方拥有各自的数字证书，该协议可以实现信息安全通信的保密性、完整性和身份可认证性。

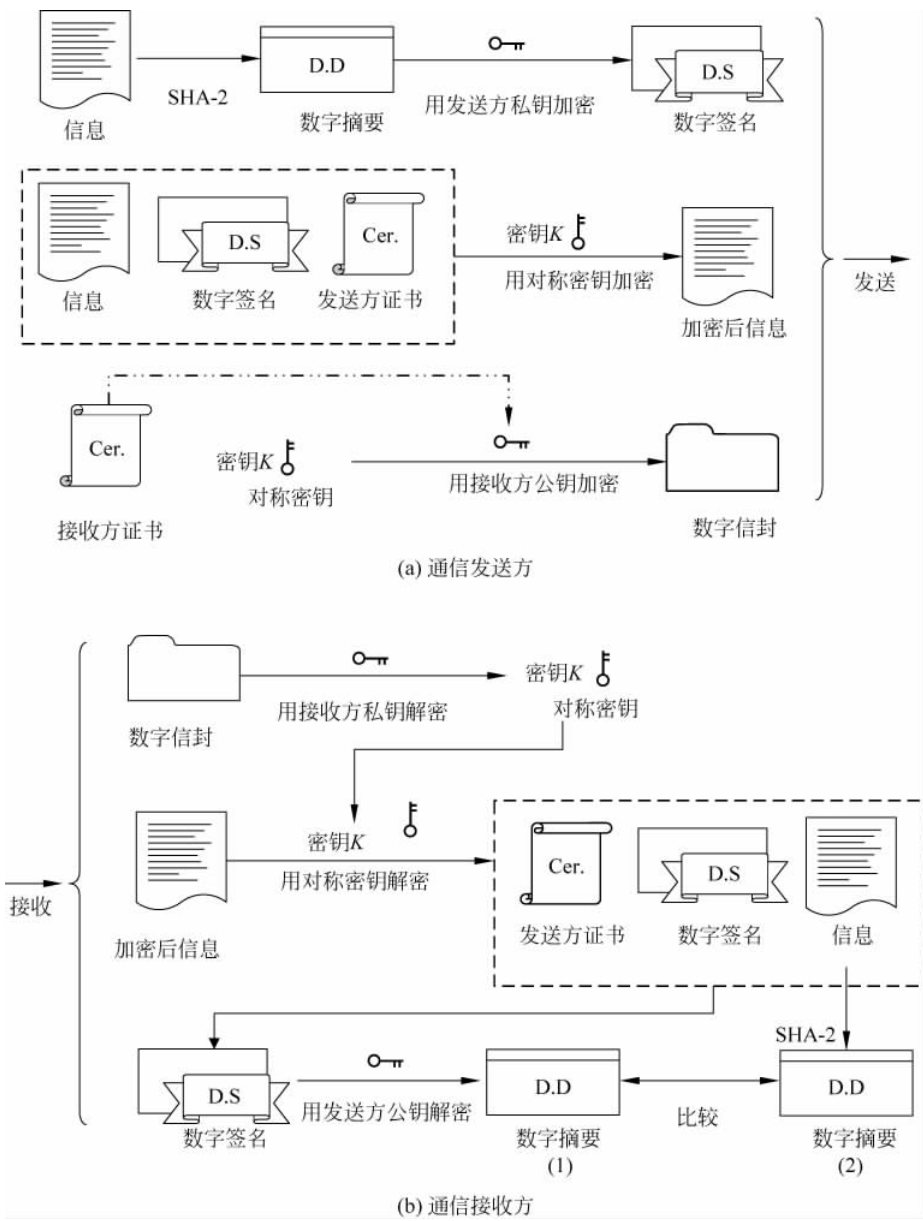

图 3-1　数字证书安全通信原理

3.2 密码学技术基础

密码学是以数学理论为基础，通过研究和设计安全、可靠的密码编码方法来实现信息安全通信的科学，主要包括密码编码和密码分析与破译。密码学的主要研究对象是密码编码，通过分析密码编码的安全性、执行效率和安全强度来评估密码编码的性能。密码学是信息安全技术的核心，是实现通信保密、数字签名和身份认证等信息安全技术的基石，也是构成各种网络安全通信协议和电子商务安全支付协议的基本组成单元。

3.2.1 密码学技术概述

密码学家 Ron Rivest 对密码学的定义是："密码学是关于如何在敌人存在的环境中通信"，这句话很明确地表达了密码学在信息安全通信中的作用。在信息安全通信协议的描述过程中经常会用到明文、密文以及加/解密等密码学名词术语，它们的具体含义是：

- 明文(Plaintext)——加密前可读的初始信息。
- 密文(Ciphertext)——加密后不可读的信息。
- 加密(Encryption)——利用加密算法将明文转换为密文的数学转换过程。
- 解密(Decryption)——利用解密算法将密文转换为明文的数学转换过程。
- 密钥(Key)——开启加密与解密的关键信息，是信息机密性的保证。
- 加密体制(Encryption System)——包括明文、密文、密钥空间以及加/解密算法。
- 密码体制(Cipher System)——加密的各种方案，包括加密体制和认证体制。
- 密码编码学(Cryptography)——密码的编码、隐藏或识别。
- 密码分析学(Cryptanalysis)——通过密码分析实现伪造或破译。
- 密码学(Cryptology)——密码编码学和密码分析学的统称。

信息安全通信的发送方使用加密算法对明文信息进行加密得到密文，即将可读的信息转换为无序的不可读的信息，接收方收到密文信息后使用相应的解密算法对密文信息进行解密得到明文，即将不可读的信息还原成原来的信息，保证了信息在公网传输过程中的保密性，其中加密和解密就是密码学算法的具体应用，如图 3-2 所示。

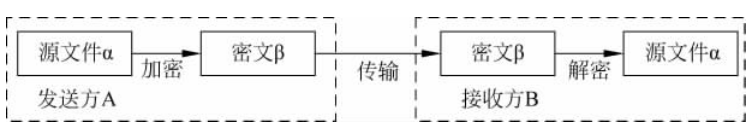

图 3-2 信息安全通信的加密与解密

信息安全通信协议描述过程中习惯用五元组(P,C,K,E,D)来描述协议各组成元素，它们的主要描述方式为：

- 用 P 来描述系统中所有明文的集合，将其称为明文空间。
- 用 C 来描述系统中所有明文对应的密文的集合，将其称为密文空间。
- 用 K 来描述系统中所有的密钥的集合，将其称为密钥空间。
- 对于任意的密钥 $k\in K$，用 $c=E_k(m)$ 来表示加密过程，用 $m=D_k(c)$ 来表示解密过

程，其中 $m \in P, c \in C$。

随着网络技术的发展，密码学已经应用于政治、经济和生活的各个领域中，并逐步发展为与数学、计算机科学、电子通信、微电子等技术相结合的交叉性科学。基于密码学的信息安全技术已经被广泛地应用于信息安全通信系统的各个环节，例如电子商务安全交易、电子政务、电子银行和电子证券等网上业务。

1. 算法的分类

按照历史发展阶段可以将加密算法划分为古典密码和现代密码。古典密码是基于字符替换的密码，随着计算机技术的发展，密码破解效率不断提高，古典密码已经不具备安全保密功能，但是它代表了密码的起源。

按照加密密钥和解密密钥是否相同可以将加密算法分为对称加密算法和非对称加密算法。

- 对称加密算法是指加密密钥和解密密钥相同，此类算法也叫单密钥加密算法和私钥加密算法。
- 非对称加密算法是指加密密钥和解密密钥不同，此类算法也叫双密钥加密算法和公钥加密算法。

按照加密算法的加密模式可以将加密算法划分为序列密码和分组密码。

- 序列密码是按二进制位或字节对明文信息进行加密，也称为流密码。序列密码主要用硬件实现，例如数字电视信号的加密。
- 分组密码是加密前将明文按照固定长度进行分组，然后对每个明文分组分别进行加密，将所有密文连接起来得到最终的密文。分组密码主要用软件实现，例如 DES (Data Encryption Standard，数据加密标准)算法。

2. 古典密码

古典密码中最具有代表性的是换位密码，就是用某个字符来代替另外一个字符，实现加密的编码方法。最早的换位密码是由 Julius Caesar 发明的凯撒密码，主要原理是字母表中的每个字母用它之后的第 k 个字母代替。以循环方式连接字母表，即字母 Z 之后就跟着字母 A，比如当 $k=6$ 时：

明文为 a b c d e f g h i j k l m n o p q r s t u v w x y z

密文为 g h i j k l m n o p q r s t u v w x y z a b c d e f

如果被加密的明文信息为 University，则：

明文为 University

密文为 Atobkxyoze

3. 分组密码

分组密码是将明文信息按照固定长度进行分组，然后对每个明文分组分别进行加密处理得到对应的密文，将所有密文分组连接起来得到最终密文信息。分组密码的主要工作模式有四种，分别是电码本模式、密码块链接模式、密码反馈模式和输出反馈模式。

1) 电码本模式

ECB(Electronic Code Book，电码本)模式是典型的分组密码工作模式，首先将给定明

文 x 进行分组 $x=x_1x_2\cdots x_i$，每个分组 x_i 的长度分别为 b 比特，e_k 是分组加密算法，产生密文分组 $c_i=e_k(x_i)$，依次将 c_i 连接起来就得到密文信息 c，即 $c=c_1c_2\cdots c_i$，假设使用 DES 加密算法，每个分组的长度为 64 比特，加密过程如图 3-3 所示。电码本模式最适合处理较短的明文信息，因为相同明文分组得到的密文是相同的，当明文信息较长时容易出现重复的分组信息，这样会降低分组密码的安全性。

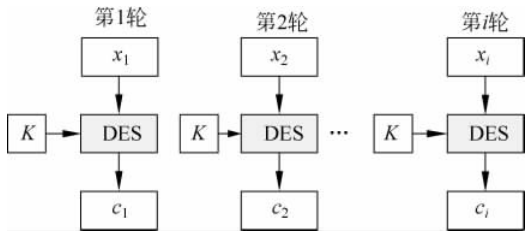

图 3-3　电码本 ECB 模式

2）密码块链接模式

CBC(Cipher Block Chaining，密码块链接)模式首先将给定明文 x 进行分组 $x=x_1x_2\cdots x_i$，每个分组 x_i 的长度分别为 b 比特，e_k 是分组加密算法，其中 IV 是一个 b 比特的初始向量。向量 IV 首先与明文的第一个分组进行按位的异或运算，将得到的结果进行加密得到对应的密文，然后该密文分组再参与下一个明文分组的加密运算。假设使用 DES 加密算法，每个分组的长度为 64 比特，如图 3-4 所示。密码块链接模式可以阻止对分组实施重放、插入和删除等攻击，但是此模式会导致错误在分组间传播。

$$
\begin{aligned}
c_1 &= e_k(x_1 \oplus IV)\\
c_2 &= e_k(x_2 \oplus c_1)\\
&\vdots\\
c_i &= e_k(x_i \oplus c_{i-1})
\end{aligned}
$$

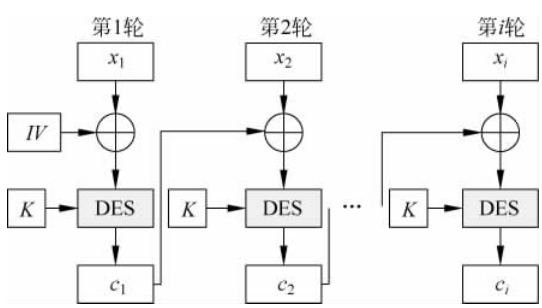

图 3-4　密码块链接 CBC 模式

3）密码反馈模式

CFB(Cipher Feedback，密码反馈)模式首先将给定明文 x 进行分组 $x=x_1x_2\cdots x_i$，每个分组 x_i 的长度分别为 b 比特，e_k 是分组加密算法，其中 IV 是一个 b 比特的初始向量。假设使用 DES 加密算法，每个分组的长度为 64 比特，首先将初始向量 IV 输入一个 64 比特的移位寄存器，加密算法输出的最左 j 位与明文的第一个分组 x_1 进行异或运算，产生密文的第一个单元 c_1，然后将移位寄存器的内容向左边移动 j 位，将前者得到的 c_1 添加到移位寄存

器的最右边，以此类推，如图 3-5 所示。密码反馈模式适合以比特或者字节为单位的数据加密，但是此模式会导致错误在分组间传播。

$$
\begin{aligned}
z_1 &= e_k(IV), \quad & c_1 &= x_1 \oplus z_1 \\
z_2 &= e_k(c_1), \quad & c_2 &= x_2 \oplus z_2 \\
&\vdots & &\vdots \\
z_i &= e_k(c_{i-1}), \quad & c_i &= x_i \oplus z_i
\end{aligned}
$$

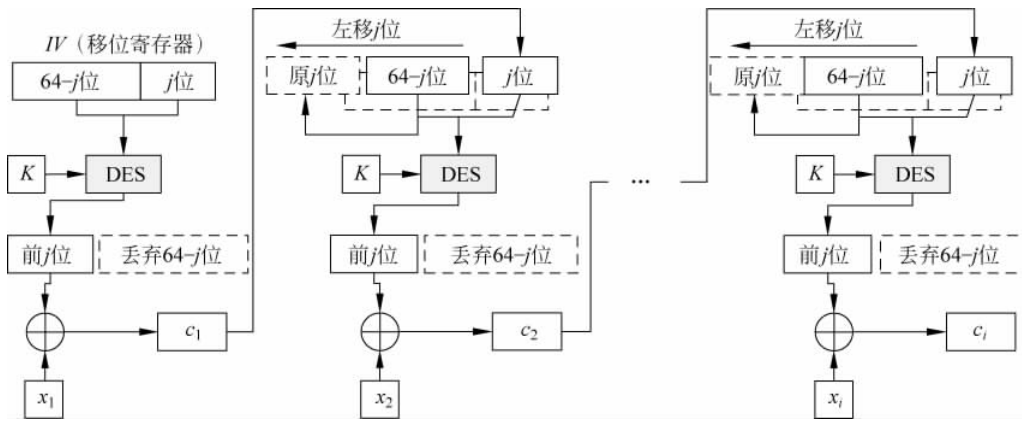

图 3-5　密码反馈 CFB 模式

4）输出反馈模式

OFB(Output Feedback，输出反馈)模式首先将给定明文 x 进行分组 $x=x_1x_2\cdots x_i$，每个分组 x_i 的长度分别为 b 比特，e_k 是分组加密算法，其中 IV 是一个 b 比特的初始向量。假设使用 DES 加密算法，每个分组的长度为 64 比特，输出反馈模式的结构类似于密码反馈模式，不同之处在于输出反馈模式是将加密算法的输出反馈到移位寄存器，而不是将密文输出反馈到移位寄存器，如图 3-6 所示。它解决了 CFB 模式中传输过程中的比特错误会被传播的问题，但是比起 CFB 模式要更加容易遭受对消息流的篡改攻击。

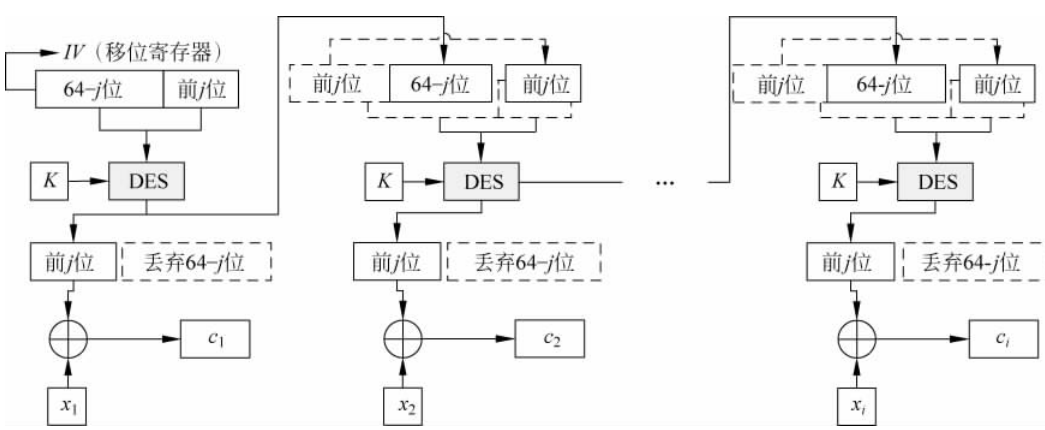

图 3-6　输出反馈 OFB 模式

$$z_1 = e_k(IV), \quad c_1 = x_1 \oplus z_1$$
$$z_2 = e_k(z_1), \quad c_2 = x_2 \oplus z_2$$
$$\vdots \qquad\qquad \vdots$$
$$z_i = e_k(z_{i-1}), \quad c_i = x_i \oplus z_i$$

通过对四种分组加密模式的具体分析，可以发现分组的大小会对分组加密的安全性产生较大影响，此外，加密算法的加密强度和密钥长短也会影响分组加密的安全性。如果分组长度过小，分组密码与古典的代替密码非常类似，系统易遭受攻击。如果分组密码的密钥长度过小，易遭受穷举攻击。此外，加密算法要具有足够的强度，才可以有效地阻止破译攻击。

4. 密码分析

密码分析是指攻击者通过破译加密算法获得密钥并还原明文的过程。密码分析者的目标并不是局限于恢复单个密文所对应的明文，而是试图恢复加密密钥，从而使得加密机制完全无效。破译者使用的分析策略取决于加密方案的固有性质以及破译者所掌握的信息数量，根据攻击者掌握信息的多少可以将攻击分为四类，如表 3-1 所示。通常有两种方法来破译传统的密码体制：一种是密码分析法；另一种是穷举攻击法。

- 密码分析法。密码分析法是通过统计和数学分析方法找到密文和明文之间的关系，利用算法的特征来推导出对应的明文，甚至是对应的密钥。密码分析法对攻击者获得的信息质量要求非常高，分析效率依赖于算法的类别和性质，以及明文的一般特征。
- 穷举攻击法。穷举攻击法是指攻击者穷举所有可能的密钥，直到发现能够解密的密钥为止。此种攻击方法非常简单，主要利用现代计算机的高速性来实现攻击。

表 3-1 密码分析的攻击类型

攻击类型	密码分析者掌握的信息
唯密文攻击	• 加密算法 • 要解密的密文
已知明文攻击	• 加密算法 • 要解密的密文 • 一个或多个密钥产生的明文-密文对
选择明文攻击	• 加密算法 • 要解密的密文 • 分析者选定的明文信息和对应的密文信息
选择密文攻击	• 加密算法 • 要解密的密文 • 分析者选定的密文信息和对应的已解密的明文信息

显然，对于攻击者来说唯密文攻击的难度最大，即攻击者所掌握的信息仅有密文和加密算法。只要加密的密钥足够长，采用穷举攻击法将会耗费相当长的时间才可以破译加密算法，因此在有限时间内是无效的。所以必须依赖对密文信息的分析，通过各种数学和统计分析方法来破译加密算法。

因为攻击者掌握的信息较少，所以唯密文攻击是最容易抵抗的。但是攻击者会采取各

种方法来获得更多的信息，通常攻击者利用各种文件特定格式来获取更多的明文和与之对应的密文信息，例如银行转账信息、C 程序以及 ASP 代码都具体固定格式的文件开头，攻击者根据这些信息可以获得更多的明文和密文对，使攻击变得更加容易。

攻击者也可以利用网络通信协议的一些漏洞来获取更多的加密信息，例如攻击者可以选择特定的明文信息诱骗网络通信协议对指定明文进行加密，这样使攻击方式转变为选择明文攻击从而明显降低了破译难度。

5. 加密体制的安全观念

判断加密体制是否安全有不同的观念，其中主要的观念有两种，即无条件安全和计算上安全。

无条件安全也称信息论安全，是指即使破译者拥有无限的计算能力和尽可能多的密文信息，但是都无法解出对应的明文和获取明文的相关信息，即使解出了，也无法验证结果的正确性。

计算上安全是指破译加密体制所付出的计算代价和时间成本超出密文信息的本身价值，并且破译密码的时间超出密文信息的有效生命期，即破译已经失去实际意义。

密码破译的速度与采用的破译方法、计算机运算速度以及信息掌握程度都有密切的关系。如果一个加密算法在计算上是安全的，那么只有采用穷举法对系统进行破译。但是，随着密钥长度的增加穷举法所需要的时间呈指数规律增加，所以在计算机速度和破译时间合理的情况下，才能采用穷举攻击方法。如表 3-2 所示，使用穷举法破译不同密钥长度的分组加密算法时所需要的时间。其中 DES 加密算法的密钥长度是 56 位，三重 DES 加密算法的密钥长度是 168 位，AES 加密算法使用的最短密钥长度是 128 位。假设普通的计算机 1μs 可以执行 1 次解密，假设大规模并行计算机 1μs 可以执行 10^6 次解密。表 3-2 的数据表明使用大规模并行计算机破译 DES 加密算法已经非常容易，因此 DES 加密算法或安全强度低于 DES 的加密算法不再是计算上安全的。

表 3-2 穷举法解密所需的平均时间

密钥长度(位)	密钥个数	1μs 执行 1 次解密	1μs 执行 10^6 次解密
26 个字符	$26!=4\times10^{26}$	$10^{26}\mu s=6.4\times10^{12}$ 年	6.4×10^{6} 年
32	$2^{32}=4.3\times10^{9}$	$2^{31}\mu s=35.8$ 分钟	2.15ms
56	$2^{56}=7.2\times10^{16}$	$2^{55}\mu s=1142$ 年	10.01 小时
128	$2^{128}=3.4\times10^{38}$	$2^{127}\mu s=5.4\times10^{24}$ 年	5.4×10^{18} 年
168	$2^{168}=3.7\times10^{50}$	$2^{167}\mu s=5.9\times10^{36}$ 年	5.9×10^{30} 年

3.2.2 对称密钥加密技术

对称密钥加密技术是指加密的密钥与解密的密钥相同，也称为常规加密、单密钥加密或私钥加密。对称加密算法具有较高的执行效率，因此在实际网络通信系统中主要使用对称加密算法来加密通信数据，但是对称加密算法也有自身的弱点，即不能解决首次对称密钥传输问题，即发送方随机产生一个对称加密密钥，但是无法通过公开网络传送给接收方，必须借助其他方法传输密钥。在 1976 年之前，即公钥加密出现之前，只有对称加密这一种加密

类型，现在对称加密依然使用很广泛。

常用的对称密钥加密算法有以下几种：

- 数据加密标准(Data Encryption Standard，DES)。
- 三重数据加密标准(Triple DES，3DES)。
- 国际数据加密算法(International Data Encryption Algorithm，IDEA)。
- 高级加密标准(Advanced Encryption Standard，AES)。
- RC4 和 RC5。

对称加密算法的基本模型如图 3-7 所示，主要由五个部分组成：

- 明文(Plaintext)——是加密前的可读信息，是加密算法的输入信息。
- 密钥(Secret Key)——开启加密与解密的关键信息，是信息机密性的保证。
- 密文(Ciphertext)——是加密后的不可读信息，是加密算法的输出信息。
- 加密算法(Encryption Algorithm)——是将明文转换为密文的数学转换过程。
- 解密算法(Decryption Algorithm)——是将密文转换为明文的数学转换过程。

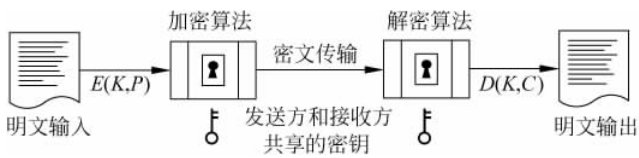

图 3-7 对称加密的基本模型

对称加密算法的速度优势使其成为数据通信的主要加密算法，因此要保证信息通信的安全性需要对称加密算法具备足够的加密强度，此外，由于对称加密算法的通信双方共享一个密钥，需要有安全的密钥共享渠道，因此对称加密要满足两个基本要求：

- 对称加密算法要具有足够的加密强度，攻击者在获得足够的明文和密文对的情况下，利用现有的计算机和数学知识在有效的时间内都无法破译加密算法，即对称加密算法可以达到计算上是安全的设计水平。
- 发送者和接收者之间建立密钥共享渠道，对称加密算法的首次通信密钥传输问题是它本身固有的弱点，因此首先在通信双方间建立一个安全通道实现密钥的传输，通常会采用公钥加密来解决对称加密的密钥传输问题。

利用对称加密算法实现保密通信的过程如图 3-8 所示。通信双方通过安全信道传输对称加密的密钥，通信双方在协商好通信密钥后，使用对称加密算法在公共网络上进行保密通信。破译者获得的基本信息包括加密算法和密文，只能在此基础上进行分析和破译。

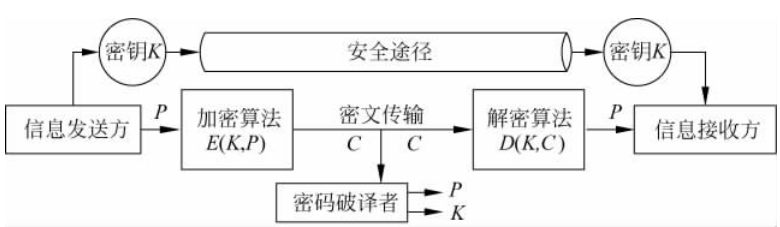

图 3-8 对称加密算法的保密通信过程

通常为描述方便，用 E 表示加密，用 D 表示解密，用 K 表示密钥，用 P 表示明文，用 C 表示密文，可以使用数学表达式来描述对称加密算法。

发送方利用密钥 K 将明文信息 P 加密得到密文 C，可以用数学表达式将加密过程表示为

$$C = E(K,P)$$

接收方利用密钥 K 将密文信息 C 解密得到明文 P，可以用数学表达式将解密过程表示为

$$P = D(K,C)$$

3.2.3 非对称密钥加密技术

非对称加密也称公钥加密，是整个密码学发展历史上最伟大的贡献。非对称加密技术是现代密码学应用的基础和核心，可以用于实现信息的保密通信和身份认证。理解公钥加密算法的基本原理需要掌握一些数论的基本概念和知识，例如素数、求余运算以及互素等概念。

1. 素数

素数是指只能被 1 和它自身整除的整数，其中 1 除外。若整数 $p>1$ 是素数，当且仅当它只有因子 ± 1 和 $\pm p$，通常加密算法和协议中只讨论非负整数。关于素数仍需注意以下几点：

(1) 对于任意整数 $a>1$ 可以被唯一地因数分解为 $a=p_1^{a_1}p_2^{a_2}\cdots p_n^{a_n}$，其中 $p_1,p_2,\cdots,p_n$ 是素数，满足 $p_1<p_2<\cdots<p_n$，且 $a_1,a_2,\cdots,a_n$ 是正整数。

(2) 素数的个数有无穷多个。

(3) 设 $\pi(x)$ 表示所有不大于 x 的素数的数目，则有

$$\lim_{x\to\infty}\pi(x)\ln x/x = 1$$

当 x 足够大时，$\pi(x)$ 可以用 $x/\ln x$ 近似表示，其中 $\ln x$ 表示自然对数。

2. 互素

设 a、b 为两个整数，将 a 和 b 的最大的公共约数称为 a、b 的最大公约数，记为 $\gcd(a,b)$，当 $\gcd(a,b)=1$ 时，称 a、b 互素。

3. 费马定理

费马定理有时被称为费马小定理，若 p 是素数，且 a 与 p 互素，则有

$$a^{p-1} \equiv 1(\mathrm{mod}\,p)$$

证明：

对于小于 p 的正整数集合 $\{1,2,\cdots,p-1\}$，用 a 乘以集合中的每个元素，并分别对 p 取模，得到集合 X。

$$X = \{a\mathrm{mod}\,p, 2a\mathrm{mod}\,p, \cdots, (p-1)a\mathrm{mod}\,p\}$$

因为 $a \neq 0$ 不能被 p 整除，所以 X 中的元素互不相等且不为 0。

模运算的基本性质：若 a 与 n 互素，且 $(a \times b) \equiv (a \times c) (\mathrm{mod} n)$，那么 $b \equiv c (\mathrm{mod} n)$。

因为 a 与 p 互素，假设存在 $ma \equiv na (\mathrm{mod} p)$ 且有 $1 \leqslant m < n \leqslant p-1$，所以根据模运算性质可以将等式两边的 a 消除，即 $m \equiv n (\mathrm{mod} p)$。但是由于 $m \neq n$ 且小于 p，因此假设不成立，X 集合中的元素都是正整数且互不相等。

X 集合中的元素互不相等，并且集合内全部整数模 p 以后的余数最多有 $p-1$ 个，分别是 $1,2,\cdots,p-1$，而集合 X 中恰好有 $p-1$ 个数，所以集合 X 中的元素模 p 以后的余数一定正好包含集合 $Y=\{1,2,\cdots,p-1\}$ 所有的元素。将集合 X 和 Y 的所有元素分别相乘，并对 p 取模，则有

$$a \times 2a \times \cdots \times (p-1)a \equiv [(1 \times 2 \times \cdots \times (p-1))] (\mathrm{mod} p)$$

$$a^{p-1}(p-1)! \equiv (p-1)! (\mathrm{mod} p)$$

根据取模运算性质，得到

$$a^{p-1} \equiv 1 (\mathrm{mod} p)$$

这里可以举个例子：设 $a=5$，$p=7$，则

$$5^6 = 5^3 \times 5^3 \equiv 6 \times 6 \equiv 1 \mathrm{mod} 7$$

$$a^{p-1} = 5^6 \equiv 1 \mathrm{mod} 7$$

费马定理的另外一种表示形式为：若 p 是素数且 a 是任意正整数，则

$$a^p \equiv a (\mathrm{mod} p)$$

请注意，这种形式下的前提条件是不要求 a 与 p 互素的。

4. 欧拉定理

首先引入数论中一个非常重要的概念，即欧拉数 $\phi(n)$。欧拉数 $\phi(n)$ 是指在区间 $[1,n]$ 中与 n 互素的正整数的个数，令 $\phi(1)=1$。

例如计算 $\phi(20)$ 的值，首先列出所有小于 20 且与 20 互素的值：

$$1,3,7,9,11,13,17,19$$

则 $\phi(20)=8$。

若 p 是素数，则 $\phi(p)=p-1$，因为区间 $[1,p-1]$ 中的所有 $p-1$ 个整数都与 p 互素。

若 p 和 q 是两个素数，且 $p \neq q$，对于 $n=pq$，则 n 的欧拉数为

$$\phi(n) = \phi(p \cdot q) = \phi(p) \times \phi(q) = (p-1) \times (q-1)$$

表 3-3 列举了 $\phi(n)$ 的前 30 个数的欧拉值。

表 3-3　前 30 个数的欧拉数

n	$\phi(n)$	n	$\phi(n)$	n	$\phi(n)$
1	1	11	10	21	12
2	1	12	4	22	10
3	2	13	12	23	22
4	2	14	6	24	8
5	4	15	8	25	20
6	2	16	8	26	12
7	6	17	16	27	18

续表

n	$\phi(n)$	n	$\phi(n)$	n	$\phi(n)$
8	4	18	6	28	12
9	6	19	18	29	28
10	4	20	8	30	8

欧拉定理：对于任意两个整数 a 与 n，且满足 a 与 n 互素，有

$$a^{\phi(n)} \equiv 1(\mathrm{mod}n)$$

证明：

设小于 n 且与 n 互素的正整数集合 $Zn=\{x_1,x_2,\cdots,x_{\varphi(n)}\}$。

由于 $\gcd(a,n)=1$，且 $\gcd(x_i,n)=1$，对 $1\leqslant i\leqslant\varphi(n)$，则有 ax_i 与 n 互素。因此 ax_1，$ax_2,\cdots,ax_{\varphi(n)}$ 构成 $\varphi(n)$ 个与 n 互素的数，且两两不同余。

因此对于任意 x_i，$a\cdot x_i\mathrm{mod}n$ 必然是集合 Zn 的一个元素。

设集合 $S=\{a\cdot x_1\mathrm{mod}n,a\cdot x_2\mathrm{mod}n,\cdots,a\cdot x_{\varphi(n)}\mathrm{mod}n\}$。则 $S=Zn$，分别计算两个集合中各元素的乘积，有

$$ax_1\cdot ax_2\cdot\cdots\cdot ax_{\varphi(n)}\equiv x_1\cdot x_2\cdot\cdots\cdot x_{\varphi(n)}\mathrm{mod}n$$

因为，若有 x_i,x_j 使得 $ax_i\equiv ax_j\mathrm{mod}n$，则由于 $\gcd(a,n)=1$，可消去 a，从而 $x_i\equiv x_j\mathrm{mod}n$。

由于 $x_1,x_2,\cdots,x_{\varphi(n)}$ 与 n 互素，故有 $a^{\varphi(n)}\equiv 1\mathrm{mod}n$ 成立。

5. 公钥加密基础

Whitfield Diffie 和 Martin Hellman 在 1976 年首次公开提出公钥加密思想。设计思想与对称密码完全不同，公钥密码是基于数学函数而非初等替换与置换，并且不像早期密码学中的对称密码那样通信双方只使用一个共享密钥，公钥密码是非对称的，使用两个完全不同并且相互独立的密钥，可以实现保密通信、密钥分发和身份认证等功能。

公钥密码学的出现解决了传统密码学中最难以解决的两个问题：密钥分发和数字签名。利用对称加密分发通信密钥主要有以下两个方法：

- 通过建立的安全传输通道传输对称加密的共享密钥。
- 利用密钥分配中心分发对称加密密钥。

建立安全的传输通道成本耗费较高，并且安全性较低，此外密钥需要经常变动以防止密钥泄露。因此，任何对称密码系统的强度都与密钥分发方法有密切联系。利用密钥分配中心分发密钥增加了密钥协商的复杂性，同时增加了通信系统的额外开销，另外，建立密钥分配中心有悖于密码学的精髓——保密性，公钥密码发明人 Whitfield Diffie 和 Martin Hellman 就曾经提到："如果必须要求用户与 KDC(Key Distributing Center，密钥分配中心)共享他们的密钥，这些密钥可能因为盗窃或索取而被泄露，那么设计不可破的密码体制究竟还有什么意义呢?"

公钥密码学的出现解决了数字签名问题，利用公钥加密体制可以实现网络通信数据的数字签名与数字认证，可以验证通信数据发送方的身份。目前，公钥加密体制已经成为网络通信实体身份认证的基础，广泛地应用于军事、政治和经济领域。

6. 公钥加密基本模型

公钥加密和认证的基本模型如图 3-9 所示，主要由六个部分组成，分别是明文、密文、加密算法、解密算法以及公钥与私钥。

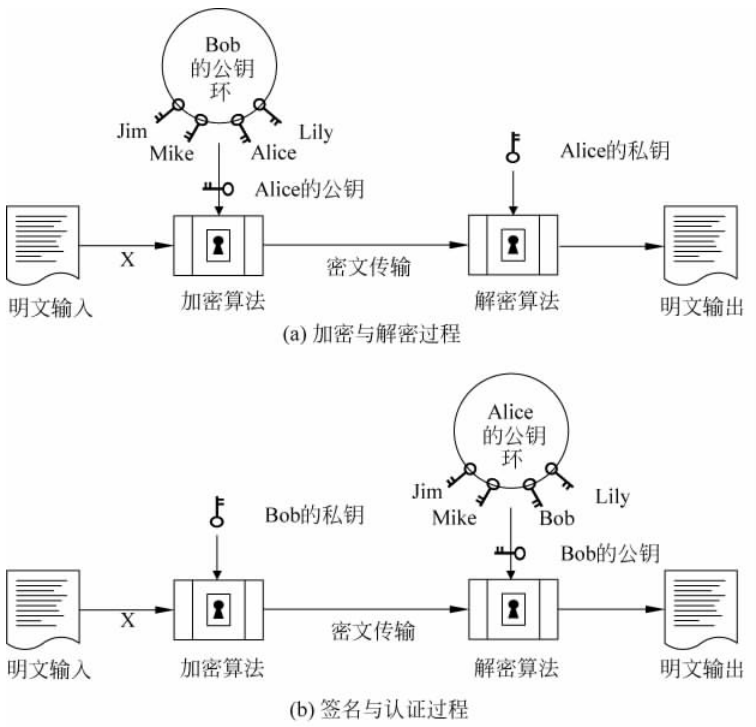

图 3-9　公钥加密和认证的基本模型

公钥加密算法的基本操作步骤如下：

(1) 每个用户随机产生一对密钥对，一个是公钥，即公开密钥，可以被其他用户下载，一个是私钥，即私有密钥，只有所有人知悉。密钥对可以用来实现数据加密与解密和数字签名与认证。

(2) 每个密钥对结合使用，用公钥加密的信息必须用私钥解密，同理用私钥加密的信息必须用公钥解密。

(3) 若 Bob 要发消息给 Alice，那么 Bob 下载并使用 Alice 的公钥对消息进行加密得到密文，并将密文发送给 Alice。

(4) Alice 收到密文后，使用自己的私钥对密文进行解密，得到明文。因为 Alice 的私钥只有自己知悉，所以只有 Alice 才能解密密文。

公钥加密的数字签名与认证的基本操作步骤如下：

(1) 若 Bob 要发消息给 Alice，并向 Alice 证明此信息确实是他发给 Alice 的，那么 Bob 首先用自己的私钥对信息进行加密，得到的密文就是 Bob 的数字签名，然后将密文发送给 Alice。

(2) Alice 收到密文后，使用 Bob 的公钥对密文进行解密，对得到的明文进行验证就是

数字认证。数字签名与数字认证是两个密切结合的过程，没有签名就没有认证。

对于公钥加密体制，任何参与网络通信的用户之间都能很容易地获得彼此的公钥，但是，私钥是在用户的本地生成，无须分发，并且其他用户很难获得。用户可以定期更换自己的密钥对，保证密钥的安全。

7. 公钥加密算法

RSA 和 Diffie-Hellman 算法是两种应用最广泛的公钥加密算法。RSA 加密算法是 1977 年由 Ron Rivest、Adi Shamir 和 Len Adleman 在麻省理工学院开发的，并于 1978 年首次公开发表。RSA 方案因其良好的安全性使其成为被最广泛接受和应用的公钥加密算法。

RSA 加密算法的主要步骤如下：

(1) 随机选择两个大素数 p、q，且 $p \neq q$，并计算 $n = p \times q$。

(2) 求 n 的欧拉数，因为 p、q 互素，且为素数，所以 $\varphi(n) = (p-1)(q-1)$。

(3) 随机选取整数 e，并满足 $\gcd(\varphi(n), e) = 1$，且 $1 < e < \varphi(n)$。

(4) 计算 d，并满足 $d \cdot e \equiv 1 \bmod \varphi(n)$。

(5) 公钥为 $PK = \{e, n\}$，私钥为 $SK = \{d, n\}$。

(6) M 为明文信息，且 $M < n$，加密过程为 $C = M^e \bmod n$，则对应的解密过程为 $M = C^d \bmod n$。

Rivest、Shamir 和 Adleman 提出的 RSA 算法使用了乘方运算，通过分组的方式对明文信息进行加密处理，只要保证每个明文分组对应的二进制数不大于 n 即可实现加密和解密。在实际应用中，分组的大小是 i 位，其中 $2^i < n \leqslant 2^{i+1}$。

假设输入的明文 $M = 88$，则 RSA 加密的基本过程如图 3-10 所示，具体步骤如下：

选择两个素数：$p = 17$ 与 $q = 11$。

计算 $n = pq = 17 \times 11 = 187$。

计算 $\varphi(n) = (p-1)(q-1) = 16 \times 10 = 160$。

随机选取 $e = 7$，使 e 与 $\varphi(n) = 160$ 互素，且小于 $\varphi(n)$。

计算 d，满足 $ed \equiv 1 \bmod \varphi(n)$ 且 $d < 160$。可知，存在整数 k，使得 $d \cdot e = k\varphi(n) + 1$，假设 $k = 1$，$7d = 160 + 1$，所以 $d = 23$。

最终得到 RSA 加密算法的公钥为：$PK = \{7, 187\}$，私钥为 $SK = \{23, 187\}$。

- 加密过程：计算 $C = 88^7 \bmod 187$。

$$88^7 \bmod 187 = [(88^4 \bmod 187) \times (88^2 \bmod 187) \times (88^1 \bmod 187)] \bmod 187$$

$$88^1 \bmod 187 = 88$$

$$88^2 \bmod 187 = 7744 \bmod 187 = 77$$

$$88^4 \bmod 187 = 59969536 \bmod 187 = 132$$

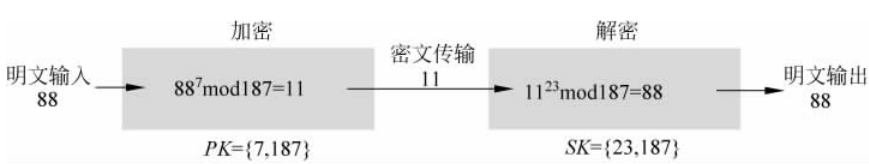

图 3-10 RSA 加密算法实例

$$88^{7}\text{mod}187 = (88 \times 77 \times 132)\text{mod}187 = 894432\text{mod}187 = 11$$

- 解密过程：计算 $M=11^{23}\text{mod}187$。

$$11^{23}\text{mod}187 = [(11^{1}\text{mod}187) \times (11^{2}\text{mod}187) \times (11^{4}\text{mod}187) \times (11^{8}\text{mod}187) \times (11^{8}\text{mod}187)]\text{mod}187$$

$$11^{1}\text{mod}187 = 11$$

$$11^{2}\text{mod}187 = 121$$

$$11^{4}\text{mod}187 = 14641\text{mod}187 = 55$$

$$11^{8}\text{mod}187 = 214358881\text{mod}187 = 33$$

$$11^{23}\text{mod}187 = (11 \times 121 \times 55 \times 33 \times 32)\text{mod}187 = 79720245\text{mod}187 = 88$$

3.2.4 两种加密技术的联合使用

公钥加密体制可以有效地解决对称加密体制的密钥分发问题，即对称加密的首次密钥传输问题。但是，公钥加密体制的运算速度远远低于对称加密体制的运算速度，并且运算过程复杂。加密和解密的过程运用了大量的幂运算，因此，在实际的网络通信过程中通信数据的加密和解密都采用效率较高的对称加密，而不采用公钥加密。基于公钥加密和对称加密各自的优点，实际的网络是联合两种加密技术来实现安全通信，如图 3-11 所示。

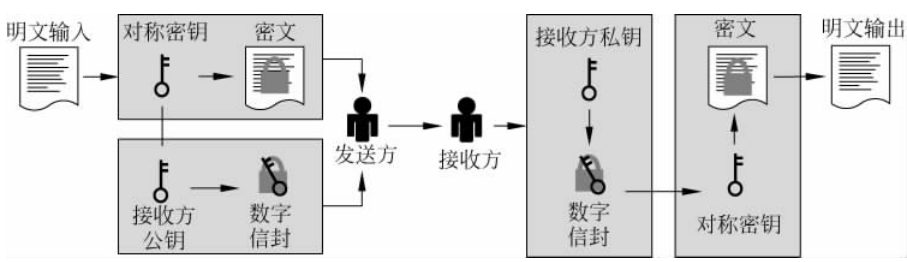

图 3-11 两种加密技术的联合使用

在如图 3-11 所示的联合加密体系中，使用对称加密算法（例如 AES 算法）加密通信数据，同时使用公钥加密算法（例如 RSA 算法）来传输对称加密算法的密钥，这样充分利用了两种算法的优点，既提高了通信数据加密和解密的速度，又解决了对称加密算法首次密钥传输的问题。

3.2.5 信息的数字摘要

数字摘要也被称为数字指纹，是将任意长度的明文信息转换为很短的固定长度的摘要信息，不同的明文信息得到不同的数字摘要。数字摘要算法，也称 Hash 函数，通常用 $H(\ )$ 表示 Hash 函数，x 表示明文信息，则对应的数字摘要值为 $H(x)=y$。Hash 函数的安全性能由发生碰撞的概率大小来衡量。如果攻击者能够轻易地构造出两个具有相同的 Hash 值的不同信息，那么这样的 Hash 函数是不安全的。

密码学的 Hash 函数必须满足下列特征：

(1) 压缩性——对于任意大小的明文输入 x，可以输出较短的固定长度的 Hash 值。

(2) 高效性——对于任意大小的明文输入 x，容易用软件或硬件求 $H(x)=y$ 的值。

(3) 单向性——对于任意给定的明文信息 x 求 $H(x)=y$ 容易，但如果任意给定 y，求 x 满足 $H(x)=y$ 在计算上是不可行的，即求 Hash 函数的逆很困难。

(4) 弱抗碰撞性——对于任意给定的明文信息 x，找到 $x'\neq x$，满足 $H(x)=H(x')$ 在计算上是不可行的，就称 Hash 函数具有弱抗碰撞性。

(5) 强抗碰撞性——寻找任意的 (x,x') 对，满足 $H(x)=H(x')$ 在计算上是不可行的，就称 Hash 函数具有强抗碰撞性。

数字摘要算法通常用来实现通信数据的完整性校验，具体过程如图 3-12 所示。

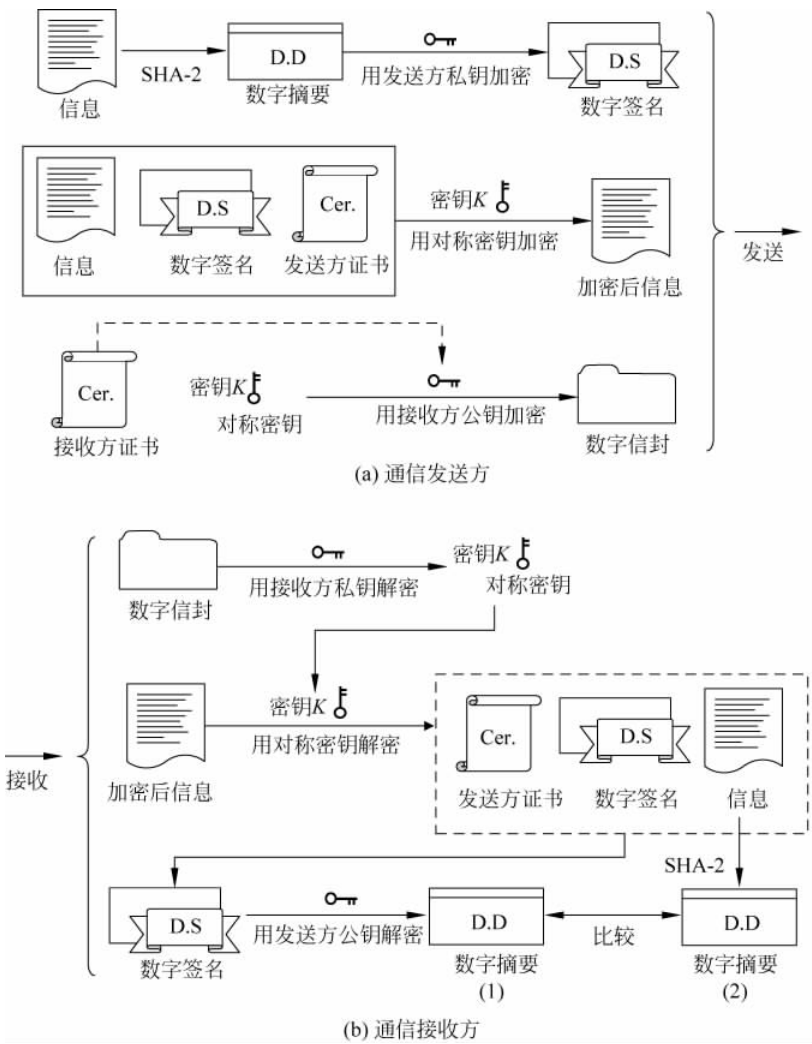

图 3-12　数字摘要算法在安全通信中的应用

发送方：

(1) 计算明文信息的数字摘要，使用发送方的私钥对数字摘要信息签名，得到数字签名。

(2) 使用对称加密算法加密明文信息、数字签名和发送方的数字证书，并将加密后的密文信息发送给接收方。

(3) 使用接收方公钥加密对称加密算法的密钥 K,形成数字信封,并将数字信封发给接收方。

接收方:

(1) 使用接收方的私钥解密数字信封,获得对称加密密钥 K。

(2) 使用对称加密密钥 K 对发送方发送的密文进行解密,得到明文信息、数字签名与发送方数字证书,并使用发送方公钥对数字签名信息进行解密,得到发送方计算的数字摘要。

(3) 接收方计算明文信息数字摘要,并与发送方计算的数字摘要比较,如果相等说明数据是完整的,否则数据在传输过程中出现错误。

3.3 常用加密标准和算法

密码学中的对称加密算法、公钥加密算法和数字摘要算法被广泛地应用于信息安全通信,其中对称加密算法的典型代表是 DES 算法,公钥加密算法的典型代表是 RSA 算法,数字摘要算法的典型代表是 MD5 算法和 SHA 算法。公钥加密算法、对称加密算法和数字摘要算法是构成网络通信协议的基础单元,可以实现通信数据的保密性、完整性和身份可认证性。

3.3.1 数据加密标准 DES

1. Feistel 密码结构

目前基于分组的对称加密算法主要源于 Feistel 分组密码结构,其算法结构是由 IBM 公司的 Horst Feistel 在 1973 年提出,因此,研究 Feistel 的密码设计原理是学习对称加密算法的基础。

根据分组方式可以将分组加密算法分为流密码和分组密码,流密码是以一个二进制位为分组单位对明文信息进行加密。而分组密码是将明文按照固定长度进行分组,然后依次对每个明文分组进行加密得到密文,最后将所有密文分组连接起来得到最终密文。目前,分组密码得到了较广泛的应用,主要的对称密码都采用分组加密模式,Feistel 对称加密算法的结构如图 3-13 所示。

Feistel 加密算法的明文分组和密钥 K 的长度是 $2w$ 位,将明文分组均分为左右两部分 L_0 和 R_0,利用第一轮的 L_0 和 R_0 计算第二轮的 L_1 和 R_1,经过 n 轮的计算得到长度为 $2w$ 位的密文。在每一轮的计算中,都有对应的子密钥参与运算,由主密钥 K 产生 n 个子密钥。

每轮加密算法相同,首先将 R_0 与 K_1 作为输入数据,经轮函数 F 运算后得到的输出数据与 L_0 进行异或运算,得到的结果作为下一轮的右半部 R_1,另外,R_0 直接作为下一轮的左半部分 L_1,依此算法完成 n 轮计算得到密文。

Feistel 分组密码结构是对称加密算法的设计基础,其具体实现依赖于以下参数:

- 分组大小——在其他条件不变的情况下,对称加密算法的分组越长加密算法的安全性越高,但是同时会降低加密和解密的速度。早期的对称加密算法的分组长度一般

为 64 位，但是随着计算机速度的提高，目前越来越多的分组密码选择 128 位的分组长度。

- 密钥大小——密钥越长安全性越高，但是同时会降低加密和解密速度，目前对称加密算法普遍使用的密钥长度是 128 位。
- 迭代轮数——对称加密算法的迭代轮数越高安全性越高，但是轮数越高加解和解密的速度越低，通常使用的迭代轮数为 16 轮。
- 子密钥产生算法——通常子密钥产生的算法越复杂，密码的分析和破译就越困难。
- 轮函数——轮函数的复杂度越高，密码分析和破译的难度就越大。

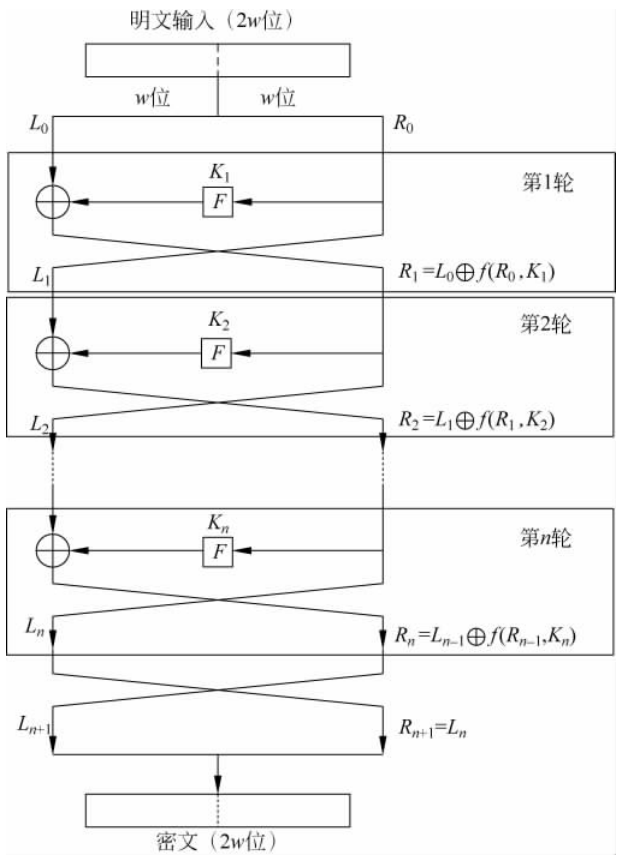

图 3-13　Feistel 密码结构

2. 数据加密标准

对称加密体制的经典算法是数据加密标准 DES 算法，于 1977 年被原美国国家标准局，即现在的美国国家标准与技术研究院 NIST 采纳为联邦信息处理标准。

DES 算法的设计是基于 Feistel 分组密码结构。在 20 世纪 60 年代后期，由 Horst Feistel 负责设计 LUCIFER 算法，LUCIFER 算法被应用于 IBM 公司的现金发放系统。LUCIFER 算法采用了 Feistel 分组密码结构，其分组长度为 64 位，密钥长度为 128 位。基于 LUCIFER 算法 IBM 公司启动商用的对称加密算法，由 Walter Tuchman 和 Carl Meyer

设计出适合单片机运行环境的对称加密算法，算法密钥为 56 位，明文分组为 64 位，在 1973 年被美国国家标准局 NBS 采纳为数据加密标准，即 DES 算法。

DES 算法的安全性一度成为信息安全界争议的焦点，一方面 DES 算法的密钥长度从 128 位降低到 56 位，无法抵御穷举攻击，而另一方面 DES 算法内部的 S 盒设计被官方列入机密，使用户无法掌握 DES 内部结构设计的安全性。即便如此，DES 算法还是得到广泛地应用，如 Internet 上的安全通信，尤其是金融行业。在 1994 年 NIST 重新决定将 DES 算法的联邦试用期延长 5 年，直到 1999 年 NIST 才宣布 DES 只能使用在早期的系统中。

3. DES 加密算法

DES 加密算法的分组长度为 64 位，密钥长度为 64 位，具体过程如图 3-14 所示。实际上 DES 加密算法的密钥只有 56 位，其余的 8 位是校验位。由 56 位的主密钥通过子密钥生成算法产生 16 个 48 位的子密钥，经过 16 轮的运算将 64 位的明文分组转换成 64 位的密文分组。DES 解密过程与加密过程基本类似，将密文作为 DES 算法的输入，经过 16 轮与加密过程逆向的解密运算，得到 64 位的明文，但是在解密过程中子密钥 K_i 的使用顺序与加密的时候相反，即第一轮解密使用 K_{16}，第二轮使用 K_{15}，以此类推，最后一轮使用 K_1。

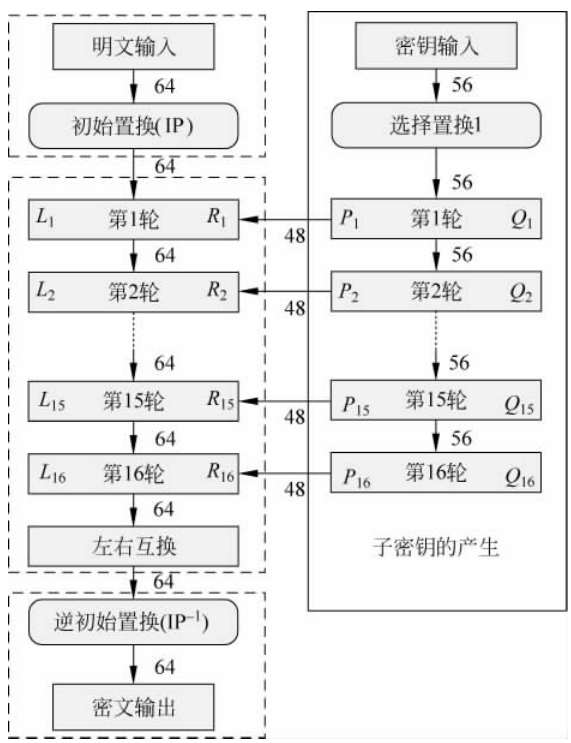

图 3-14 DES 加密算法

如图 3-14 所示，DES 加密算法首先将 64 位的明文分组进行初始置换，然后均分为左右两个部分。接着对分组信息做 16 轮的加密计算，并将最后一轮的输出数据做逆初始置换输出最终的 64 位密文分组。与 Feistel 加密比较 DES 加密的初始和结束环节增加了初始置换和逆初始置换，其余算法结构与 Feistel 结构完全一致。

DES 加密算法的密钥长度为 64 位，除去 8 位的校验位，剩余 56 位的加密密钥，经过 16 轮的迭代产生 16 个子密钥 $K_1, K_2, \cdots, K_{16}$，并参与每轮的加密计算。

下面详细讲解 DES 加密算法的执行过程。

1）初始置换与逆初始置换

DES 加密算法的初始变换是按照 IP 变换表对 64 位的明文序列进行位置的调换，调换前对每个位进行标记，IP 变换表标识了每个位调换后的次序，如表 3-4 所示。

表 3-4 IP 变换表

58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

DES 加密算法的逆初始变换是按照 IP^{-1} 变换表对经过 16 轮加密运算的 64 位的密文输出序列进行位置的调换，调换前对每个位进行标记，IP^{-1} 变换表标识了每个位调换后的次序，如表 3-5 所示。

表 3-5 IP^{-1} 变换表

40	8	48	16	56	24	64	32	39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30	37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28	35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26	33	1	41	9	49	17	57	25

2）每轮加密过程

DES 加密算法的每轮迭代计算过程，如图 3-15 所示。首先 64 位的明文分组经过初始变换后均分为左边 32 位和右边 32 位，分别使用 L_1 与 R_1 表示。上一轮的数据输出作为下一轮的数据输入，上一轮得到的两个数据分别用 L_{i-1} 与 R_{i-1} 表示。首先对 R_{i-1} 进行 F 函数运算，将输出的 32 位结果与 L_{i-1} 进行异或运算，然后将运算结果作为下一轮的右半部分 R_i，最后将 R_{i-1} 直接作为下一轮的左半部分 L_i。

DES 加密算法的 F 函数运算过程如图 3-15 所示，首先通过扩张置换将 R_{i-1} 从 32 位扩展到 48 位，扩展置换表如表 3-6 所示，扩展算法的基本原理是将原数据的其中 16 位重复一次插入数据表中。

表 3-6 扩展置换表

32	1	2	3	4	5	4	5	6	7	8	9
8	9	10	11	12	13	12	13	14	15	16	17
16	17	18	19	20	21	20	21	22	23	24	25
24	25	26	27	28	29	28	29	30	31	32	1

扩展后的 48 位数据与密钥 K_i 做异或运算得到 48 位数据输出，并通过 S 盒运算后输出 32 位结果值，S 盒的工作原理如图 3-16 所示，S 盒由 8 个子盒组成。

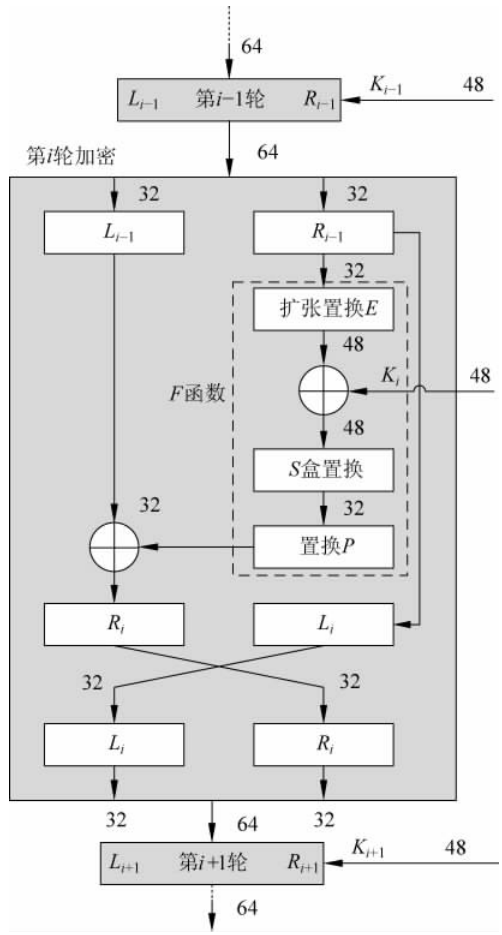

图 3-15 DES 每轮迭代计算过程

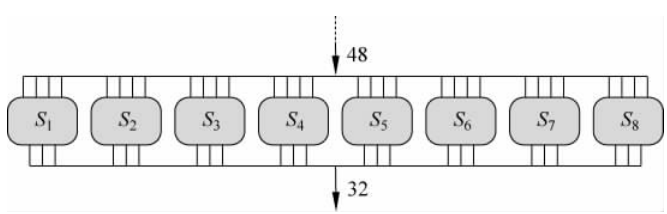

图 3-16 S 盒的内部结构

首先 S 盒将 48 位的输入数据均分为 8 份，每个子盒输入数据为 6 位，经过子盒处理后得到 4 位输出，其中 S 盒的第一个子盒 S_1 的变换表如表 3-7 所示。

表 3-7 子盒 S_1 的变换表

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_1 盒的输入为 6 位，将第 1 位和第 6 位组成的二进制数作为行号，将中间 4 位组成的二进制数作为列号，根据行号和列号查询 S_1 的变换表，将查询得到的十进制数转化为 4 位二进制数，并将此 4 位二进制数作为 S_1 盒的输出，其中行号和列号的范围分别为 0～3 和 0～15。

假设 S_1 盒的输入值为 111001，那么行号为二进制数 11，列号为二进制数 1100，查询 S_1 的变换表得到的十进制数据为 10，所以 S_1 盒的输出值为 1010。

S 盒的输出数据为 32 位，将经过 P 置换后的 32 位数据与 L_{i-1} 进行异或运算，并将异或运算结果作为下一轮的右半部分 R_i，其中 P 置换过程如表 3-8 所示。

表 3-8　P 置换表

16	7	20	21	29	12	28	17	1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25

3）密钥生成过程

DES 加密算法的密钥产生过程如图 3-17 所示。

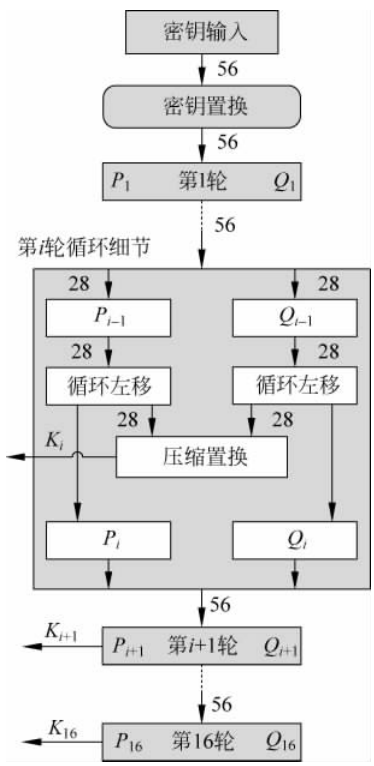

图 3-17　DES 加密算法的密钥产生过程

DES 加密算法的密钥为 64 位，其中每隔 8 位就提供一个数据校验位，因此，去掉 8 位校验数据后，实际 DES 加密算法的密钥长度为 56 位。首先对 56 位的密钥实施密钥置换，密钥置换过程如表 3-9 所示。

表 3-9 密钥置换

57	49	41	33	25	17	9	1	58	50	42	34	26	18
10	2	59	51	43	35	27	19	11	3	60	52	44	36
63	55	47	39	31	23	15	7	62	54	46	38	30	22
14	6	61	53	45	37	29	21	13	5	28	20	12	4

将置换后的 56 位密钥均分为 P 和 Q 两个部分，分别对 P 和 Q 循环左移，在每轮的密钥迭代计算过程中，上轮输出的 P_{i-1} 和 Q_{i-1} 分别经过循环左移一位或两位得到下轮的 P_i 和 Q_i。产生每轮密钥时 P 和 Q 的循环左移的位数如表 3-10 所示。

表 3-10 每轮密钥循环左移位数

轮数	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
移位次数	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

对每轮得到的 P_i 和 Q_i 进行压缩置换，得到 48 位的输出数据，将其作为子密钥 K_i 参与 F 函数运算，压缩置换如表 3-11 所示。

表 3-11 压缩置换

14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

3.3.2 其他对称密钥加密算法

1. 三重 DES

随着计算机运算速度的提高，DES 加密算法在穷举攻击之下越来越显脆弱，因此提高对称加密算法的加密强度成为必然趋势。一种思路是设计新的对称加密算法来替代 DES 算法，例如 AES 算法；另一种思路是改进 DES 加密算法，增加 DES 的加密次数和密钥数量，例如三重 DES 算法。

美国国家标准与技术研究院 NIST 于 1999 年将三重 DES 指定为过渡的加密标准，三重 DES 加密算法使用了 3 个密钥并执行 3 次 DES 算法，通过加密-解密-加密的顺序来执行加密过程，通过解密-加密-解密的顺序来执行解密过程，如图 3-18 所示。

三重 DES 可以使用三个不同的密钥，其有效密钥长度达到了 168 位，也可以在第一步和第三步使用相同的密钥，即 $K_1=K_3$，此时有效密钥长度为 112 位。密钥长度为 168 位的三重 DES 算法可以有效地抵抗穷举攻击，可以作为 DES 的替代算法，但是三重 DES 运算会大幅降低加密和解密的执行效率。

2. 高级加密标准

三重 DES 算法没有从根本上解决 DES 加密算法的脆弱性，因为加密和解密依赖于 DES 加密算法，三重 DES 和 DES 加密算法的明文分组都为 64 位，从安全性考虑需要设计

更大的明文分组的对称加密算法，并不能将三重 DES 视为能够长期使用的对称加密算法。

美国国家标准与技术研究院 NIST 于 1997 年公开征集 AES（Advanced Encryption Standard，高级加密标准），要求新算法在安全性和执行效率上都要优于三重 DES 加密算法。在 2001 年将 Joan Daemen 和 Vincent Rijmen 所设计的 Rijndael 算法作为 AES 加密算法。

Rijndael 加密算法具有如下特性：

- 对于所有已知的安全攻击具有免疫性。
- 适合于各种平台，执行效率高。
- 加密算法设计过程简单，程序代码紧凑。

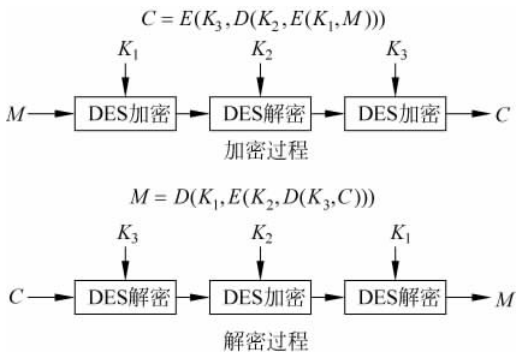

图 3-18　三重 DES 加密算法

实际 AES 加密算法和 Rijndael 加密算法之间有一定的区别，对于 Rijndael 加密算法，分组长度和密钥长度都可以被指定为 128 位、192 位或 256 位，而对于 AES 加密算法，密钥的长度可以选择 128 位、192 位或 256 位，但是分组长度只能够使用 128 位。AES 加密算法的相关参数与密钥长度有关，本书讲解 AES 算法时选择应用相对广泛的 128 位密钥，不同密钥长度的 AES 加密算法的各参数如表 3-12 所示。

表 3-12　AES 加密算法的参数

密钥长度（字数/字节数/位数）	4/16/128	6/24/192	8/32/256
分组长度（字数/字节数/位数）	4/16/128	4/16/128	4/16/128
轮数	10	12	14
每轮密钥长度（字数/字节数/位数）	4/16/128	4/16/128	4/16/128
扩展密钥长度（字数/字节数）	44/176	52/208	60/240

AES 加密算法的结构不同于 Feistel 加密算法结构，没有像 Feistel 算法那样将明文分组分割为左右两部分，加密算法的轮数也不相同。AES 加密算法的主要特点是用一部分数据来处理另一部分数据，在每轮替换和移位计算的时候同时并行处理整个数据分组，AES 加密算法的加密和解密过程如图 3-19 所示。

当密钥为 128 位时，AES 加密算法的加密和解密的轮数为 10 轮。加密时每轮迭代包含四个步骤，分别是字节替换、行移位、列混合和轮密钥加，最后一轮只包含三个步骤。同样，解密时每轮迭代包含四个步骤，每个步骤是加密的逆运算，最后一轮只包含三个步骤。AES 加密算法主要有如下四个步骤：

- 字节替换——是以字节为单位的非线性变换，利用 AES 加密算法的 S 盒将加密数据的每个字节用另外一个字节代替。

- 行移位——是以字节为单位的线性变换，对加密数据实施行内的循环移位。
- 列混合——是以字节为单位的线性变换，利用列变换函数对数据实施以列为单位的变换。
- 轮密钥加——是将正在处理的明文分组数据与每轮的轮密钥进行按位的异或运算。

AES 加密是在密钥的作用下实现加密数据的扩散、混乱和非线性转换，将明文分组转换为密文分组。对于字节替换、行移位、列混合和轮密钥加运算，在解密算法中都有相应的逆运算，但是 AES 的解密算法结构和加密算法结构不相同，这一点由 AES 的算法结构决定。

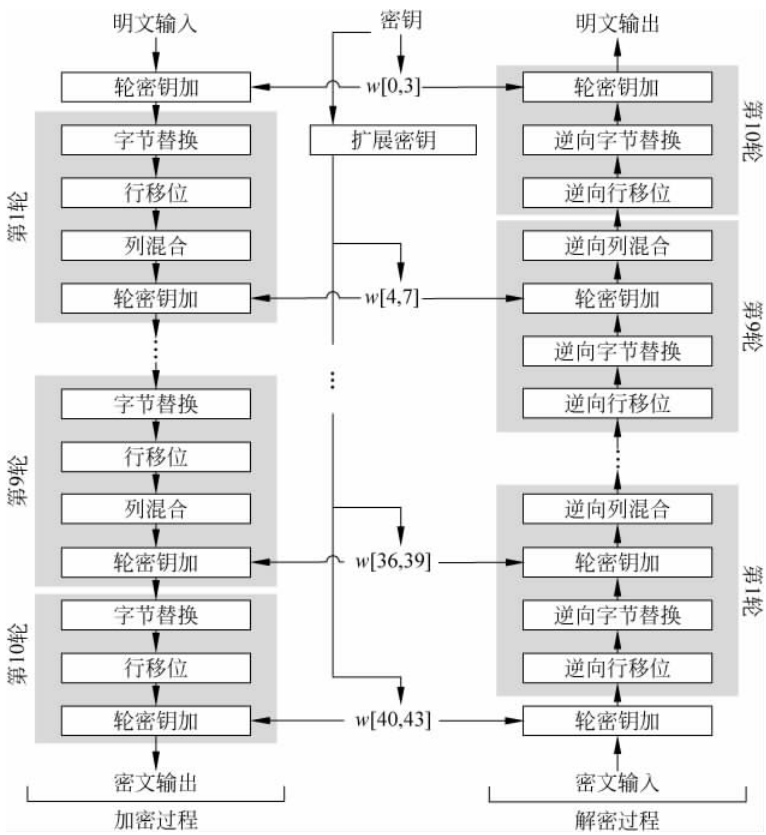

图 3-19 AES 加密算法的执行过程

AES 加密算法输入的明文分组是 128 位，每个明文分组用以字节为单位的正方形矩阵来描述，如图 3-20 所示，每个 128 位的明文分组转换为 16 字节，按照从上到下、从左至右的方式进行排列，形成一个 4×4 的矩阵。每个明文分组经过重复的替换和移位运算得到 128 位的密文分组，将所有密文分组连接起来得到最终密文，同理，解密时将 128 位的密文分组输入转换为 4×4 的矩阵并进行解密运算输出明文分组。

选择 AES 加密算法的密钥长度为 128 位，如图 3-21 所示，将 128 位的密钥转换为以字节为单位的 4×4 矩阵，通过密钥扩展算法将密钥扩展成一个以字为单位的密钥序列数组，其中每个字由 4 个字节组成。密钥序列的长度等于加密算法的轮数加 1 后与明文分组的长度相乘，例如 AES 加密算法的密钥长度为 128 位，共经历 10 轮运算，加上起始的一次轮密钥加，所以经过扩展后成为 11×128＝1408 位的密钥序列，其中密钥矩阵中的字节依旧是按

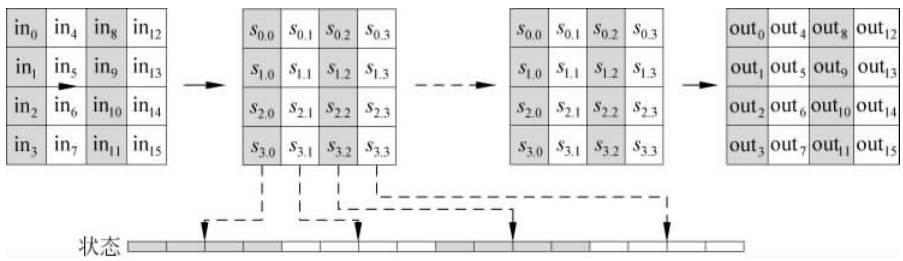

图 3-20　AES 加密算法的明文分组

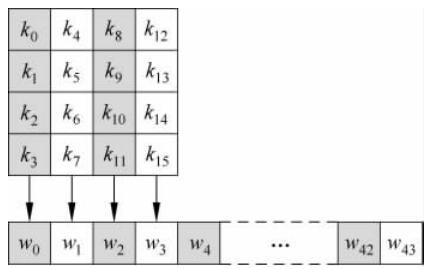

图 3-21　密钥与扩展密钥

照从上到下、从左至右的顺序排列。

AES 加密算法的结构与 Feistel 加密算法的结构不同，它每轮计算中都使用替换、移位和混乱来并行处理整个分组，因而它扩散和混乱的速度比 DES 加密算法快。AES 加密算法每轮的迭代加密过程如图 3-22 所示。

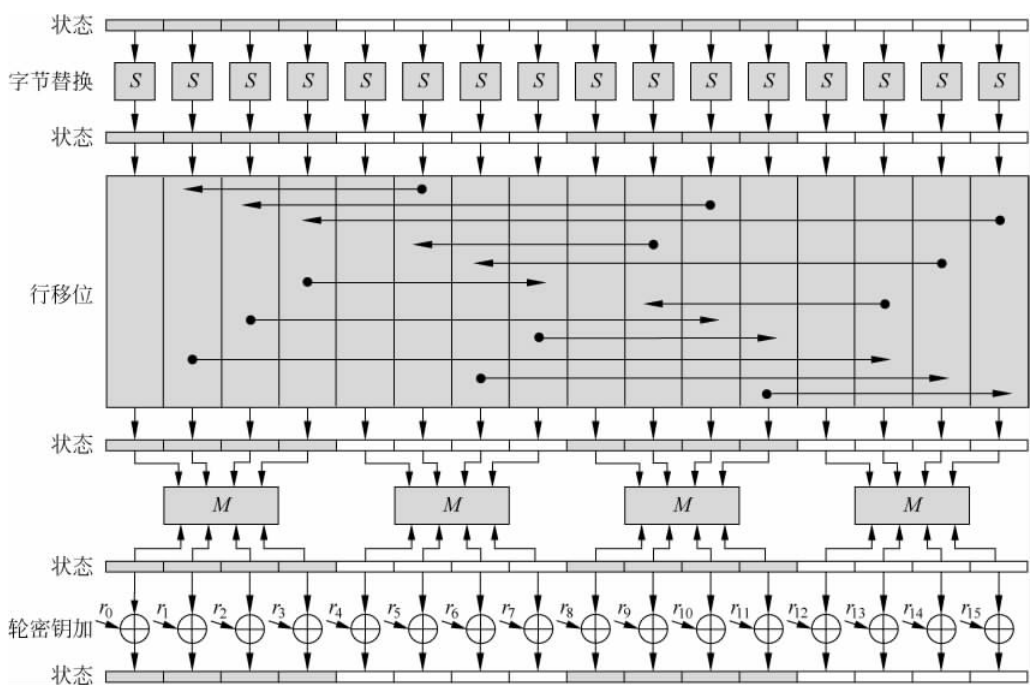

图 3-22　每轮的迭代加密过程

3. RC4 加密算法

RC4 加密算法是由 RSA Security 公司的 Ron Rivest 于 1987 年提出的密钥长度可变的流密码。流密码也称序列密码，是以二进制位或字节为单位的加密算法，通常流密码是以字节为单位加密明文信息，其主要特点是便于硬件实现、加解密运算速度快、没有或只有有限的错误传播。流密码加密时，首先将密钥输入伪随机数发生器，利用伪随机数发生器产生一串 8 位的随机数，并对随机数与明文字节流进行异或运算得到密文字节流，如图 3-23 所示。

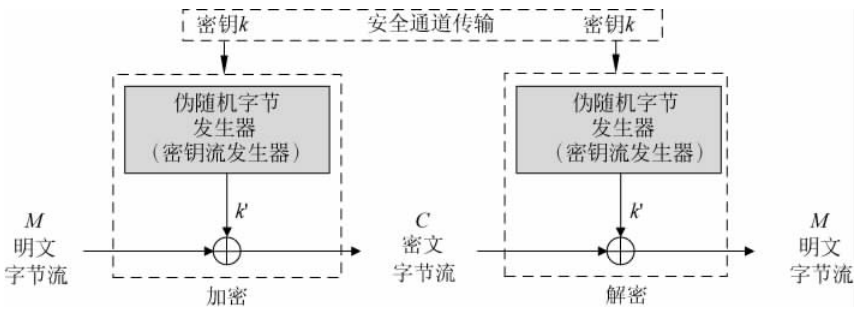

图 3-23 流密码结构示意图

在流密码加密过程中，伪随机流与明文字节流进行异或运算得到密文字节流；相反地，在解密时，伪随机流与密文字节流进行异或运算得到明文字节流。例如，假设伪随机字节发生器产生的下一字节的伪随机数为 11010101，明文为 10100011，那么异或运算后得到的密文是 01110110。

$$\begin{array}{rl} & 10100011 \quad \text{明文} \\ \oplus & 11010101 \quad \text{密钥流} \\ \hline & 01110110 \quad \text{密文} \end{array}$$

同理，解密使用相同的伪随机数序列，异或运算后得到的明文为 10100011。

$$\begin{array}{rl} & 01110110 \quad \text{密文} \\ \oplus & 11010101 \quad \text{密钥流} \\ \hline & 10100011 \quad \text{明文} \end{array}$$

影响流密码安全性的主要因素有密钥长度和密钥流的随机性。

- 密钥流序列的周期要长。密钥流生成器产生的密钥序列最终总会出现重复，重复的周期越长，密码分析的困难性越大。
- 密钥流的随机性要高。密钥流生成器是一个伪随机发生器，输出数据的随机性越高，密码分析的困难性越大。以二进制位为单位的流密码，密钥流中 1 和 0 的个数应近似相等；而以字节为单位的流密码，流密码中所有可能的 256 种字节值出现的频率应近似相等。
- 密钥要足够长。密钥作为伪随机发生器的种子值也应该有足够的长度，至少长度为 128 位，密钥越长密钥流生成器产生的密钥流的随机性越好。

如果流密码的伪随机数生成器设计的合理，在相同的密钥长度下，流密码具有与分组密码相同的安全性。但是相对于分组密码，流密码的优势在于其拥有更快的加/解密速度以及

更简洁的代码。流密码适合数据流的加密和解密，例如网络流媒体加密，而分组加密适合成块的数据加密和解密，例如文件传输、电子邮件和数据库等。

RC4 加密算法的基础是随机置换，其密码流的周期大于 10^{100} 位，平均每个输出字节仅需要 8～16 条操作指令，运行速度非常快。RC4 加密算法广泛地应用于流媒体安全通信，例如安全通信协议 SSL/TLS 就采用 RC4 算法来实现浏览器与服务器之间的安全通信，此外，RC4 加密算法已经被广泛应用于无线局域网的安全通信协议中。

RC4 加密算法主要包括初始化算法和伪随机子密码生成算法两大部分，其中初始化过程包括 S 向量的初始化和初始排列。

1）初始化 S 向量

首先给 S 向量的元素按照升序赋值，即 $S[0]=0$，$S[1]=1$，…，$S[255]=255$。同时建立临时向量 T，若密钥 K 的长度等于 256 字节时，将 K 赋给 T，若密钥 K 的长度小于 256 字节时，则使用密钥 K 重复填充 T 中的元素，直到 T 中的所有元素都被赋值为止。向量 S 和向量 T 的初始化过程如图 3-24 所示。向量 S 和向量 T 的初始化操作如下：

```
For i = 0 to 255 do
S[i] = i;
T[i] = K[i mod keylength];
```

其中 keylength 为密钥的字节数。

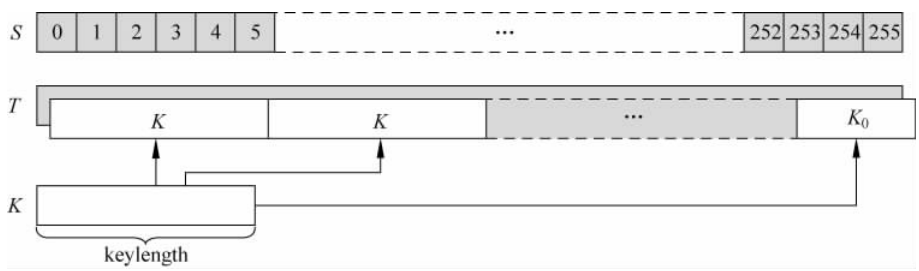

图 3-24　向量 S 和向量 T 的初始化

2）向量 S 的初始排列

根据向量 T 对向量 S 的元素位置实施互换，因为向量 T 由密钥 K 转换而来，此互换过程实际上是基于密钥 K 完成，经过互换得到新的 S 向量用于产生密钥流。向量 S 的初始排列过程如图 3-25 所示。向量 S 的初始排列操作如下：

```
j = 0;
for i = 0 to 255 do
j = (j + S[i] + T[i]) mod 256;
swap (S[i] , S[j]);
```

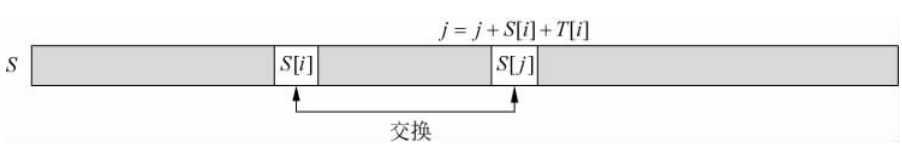

图 3-25　向量 S 的初始排列

3）密钥流的生成过程

根据当前向量 S 的值对当前 S 向量中元素的位置进行互换，并输出一个字节的流密钥，将输出的流密钥与下一个明文字节进行异或运算，得到一个字节的密文并输出。当向量 S 的最后一个元素 $S[255]$ 完成置换之后，操作重复从 $S[0]$ 开始，密钥流的生成过程如图 3-26 所示，具体操作过程如下：

```
i , j = 0;
while (true)
i = (i + 1) mod 256;
j = (j + S[i]) mod 256;
swap (S[i] , S[j]);
t = (S[i] + S[j]) mod 256;
k = S[t];
```

k 作为密钥流的输出，将 k 的值与下一明文字节进行异或运算得到密文。

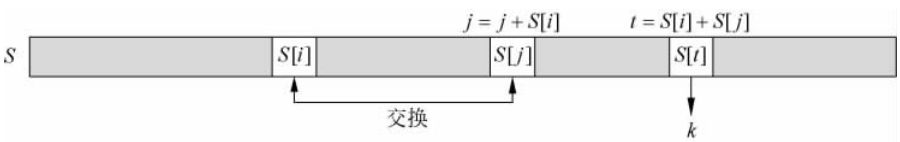

图 3-26　密钥流的生成过程

3.3.3　其他公钥加密算法

ECC(Elliptic Curve Cryptography，椭圆曲线加密)属于公钥加密体制，具有加密密钥短、安全强度高的特点，密钥长度为 164 位的 ECC 算法的安全性与密钥长度为 1024 位的 RSA 算法的安全性相当。在相同安全需求的情况下，使用 ECC 加密算法可以有效地降低密钥长度、降低系统资源占用、节约网络通信带宽和提高系统运算速度。ECC 算法尤其适合计算能力和存储能力低的移动通信终端设备，例如智能手机和掌上电脑。随着计算机运算速度的提高，要保证 RSA 加密算法的安全性，就要不断地增加密钥的长度，致使基于 RSA 的公钥加密和数字签名效率越来越低，特别是对于安全需求较高的电子银行交易系统和电子商务支付系统将产生很大的影响。因此，执行效率和安全强度均优于 RSA 加密算法的椭圆曲线加密算法已经成为公钥加密体制的主流算法，并广泛应用于网络安全通信，例如 3G 无线网络安全通信。

3.3.4　数字摘要算法 MD5

MD5(Message Digest Algorithm，消息摘要算法)也称数字摘要算法或者数字指纹，是一种单向散列函数，主要功能是实现网络通信数据的完整性校验。该算法是在 20 世纪 90 年代初由麻省理工学院的 Ronald Rivest 提出，经由 MD2、MD3 和 MD4 版本发展而来。

MD5 算法的特点是可以将任意长度的明文信息压缩为 128 位的数字摘要值，当明文信息发生微小的变化时，对应的数字摘要值都会发生剧烈的变化，因此，信息安全通信经常利用数字摘要这一特点校验通信数据的完整性。MD5 是一种单向散列函数，即由明文信息计

算数字摘要值容易，但由数字摘要值计算对应的明文信息在计算上是不可能的。

MD5 算法将明文信息按照 512 位进行分组，然后将每个分组划分为 16 个 32 位的子分组。MD5 算法经过 4 轮运算，每轮运算都要进行 16 步迭代运算，4 轮共需要 64 步完成。经过 64 步运算，最终输出 4 个 32 位的数字摘要分组，将这 4 个 32 位的数字摘要分组级联后生成 128 位的数字摘要值。

3.3.5 安全散列算法 SHA

SHA(Secure Hash Algorithm，安全散列算法)由美国国家标准与技术研究所(NIST)开发，并于 1993 年公布，通常称为 SHA-1。2002 年 NIST 发布了三个新版本的 SHA，分别是 SHA-256、SHA-384 和 SHA-512。

SHA 是一种数字摘要算法，也称数字指纹，主要功能是实现通信数据的完整性校验，可以将任意长度的明文信息压缩为固定长度的数字摘要值输出。SHA 是一种单向散列函数，即由明文信息计算数字摘要值比较容易，但由数字摘要值计算对应的明文信息在计算上是不可能的。不同版本的 SHA 算法的主要参数如表 3-13 所示。

表 3-13 不同版本 SHA 算法的参数

参数 \ 版本	SHA-1	SHA-256	SHA-384	SHA-512
摘要值长度(位)	160	256	384	512
消息长度(位)	$<2^{64}$	$<2^{64}$	$<2^{128}$	$<2^{128}$
块大小(位)	512	512	1024	1024
字大小(位)	32	32	64	64
步骤数量	80	64	80	80

SHA-512 算法可以处理长度不超过 2^{128} 位的明文信息，生成固定大小的数字摘要值为 512 位，每次处理的明文块大小为 1024 位，经过 80 个步骤的运算输出数字摘要值，数字摘要的运算过程如图 3-27 所示。

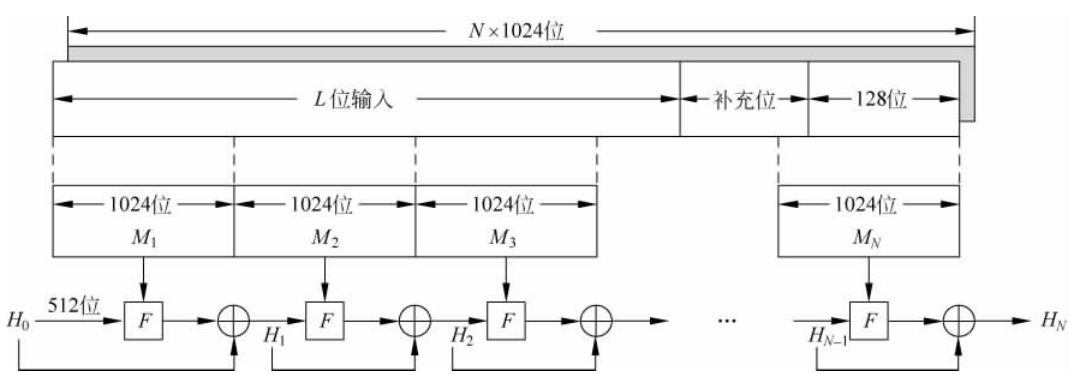

图 3-27 SHA-512 算法的运算过程

SHA-512 算法的具体步骤如下：

(1) 首先对明文信息进行填充，使填充后的明文信息长度是 1024 的整数倍少 128 比

特，可以表示为 $N\times1024+(1024-128)$位。填充部分由单个比特 1 和数个 0 组成，使其长度达到要求。

(2) 在填充后的明文信息后追加 128 比特的数据块，此数据块是 128 比特的无符号整数，主要描述明文信息的长度。

(3) 明文信息经过填充后长度为 1024 比特的整数倍，然后按照 1024 比特对其进行分割得到的数据块为 $M_1,M_2,M_3,\cdots,M_N$。

(4) 建立 512 比特的缓冲区来保存散列函数的中间值和最后结果，该缓冲区是由 8 个 64 比特的寄存器 a、b、c、d、e、f、g 和 h 组成，首先对寄存器进行初始化并分别存储一个 64 比特的整数。

$$a = 6A09E667F3BCC908 \quad e = 510E527FADE685D1$$
$$b = BB67AE8084CAA73B \quad f = 9B05688C2B6E6C1F$$
$$c = 3C6EF372FE94F82B \quad g = 1F83D9ABFB41BD6B$$
$$d = A54FF53A5F1D36F1 \quad h = 5BE0CD19137E2179$$

(5) 依次处理长度为 1024 比特的数据块 $M_1,M_2,M_3,\cdots,M_N$，每个数据块经过 80 次的迭代运算，输出 512 比特的数字摘要参与下一个数据块的迭代运算，直到处理完所有的数据块并将寄存器 a、b、c、d、e、f、g 和 h 中的数据级联后作为最终数字摘要输出，其中迭代运算过程如图 3-28 所示。

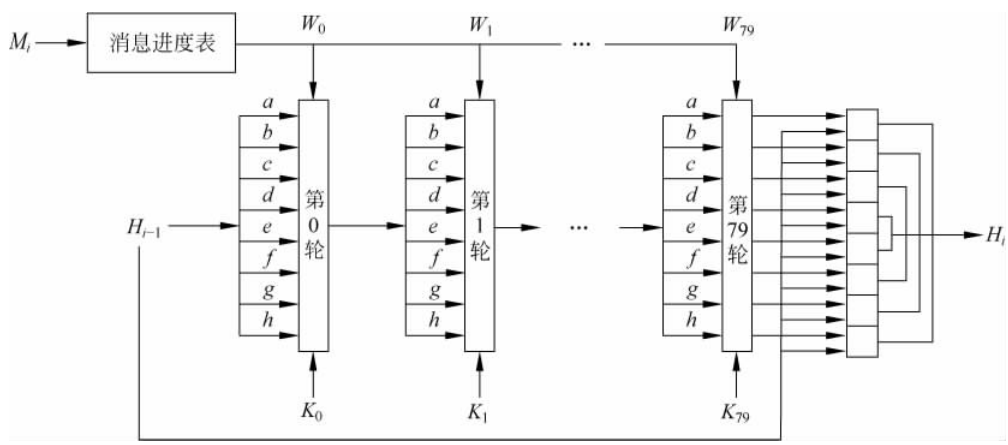

图 3-28 SHA-512 迭代运算过程

3.4 数字签名技术

数字签名与认证技术是实现网络通信实体身份认证的主要技术基础。利用数字签名与认证技术可以验证通信数据的来源和签名者的身份，并且可以实现通信数据的抗抵赖性。除了通常使用的数字签名算法外，还有一些特殊用途的数字签名算法也被广泛地应用于不同的安全通信环境，主要有盲签名算法、代理签名算法、群签名算法以及椭圆曲线签名算法等。

3.4.1 数字签名的基本原理

数字签名系统由签名者、验证者、密钥、签名算法和待签名消息等部分组成。

数字签名技术基于公钥加密体制，签名者使用自己的私钥对信息进行加密，并将加密后的密文发送给验证方，验证方利用签名者的公钥解密并验证。数字签名可以验证通信数据的真实性，可以有效地预防仿冒攻击。

数字签名的主要过程如下：

- 签名者使用单向散列函数(Hash 函数)计算明文信息的数字摘要。
- 签名者使用自己的私钥对数字摘要进行数字签名。
- 签名者发送明文信息和已签名的数字摘要给验证者。
- 验证者使用数字摘要算法计算明文信息的数字摘要值，并解密签名者签名的数字摘要，比较两个数字摘要，如果相同，则表示数据完整并身份可信；否则拒绝接收。

3.4.2 基于公钥密码体制的数字签名

基于公钥密码体制的数字签名可以实现身份认证和数据完整性校验。签名者首先利用自己的私钥对明文信息加密，然后将密文作为数字签名，连同相应的明文信息一起发送给验证者。验证者利用签名者的公钥对密文进行解密，并与签名者发送的明文信息进行比较以验证签名。

1. RSA 签名算法

签名者利用自己的私钥对明文信息加密，并将密文作为数字签名与明文信息一同发送给验证者。验证者利用签名者的公钥对密文进行解密并验证签名的有效性。同时，为了保证数字签名的时效性，通常签名者会在明文信息后附加时间标记来防止重传攻击。

2. El Gamal 签名算法

El Gamal 签名算法的安全性是基于求解有限域上离散对数的困难性。

首先选择一个大素数 p，整数 g 是 Z_p^* 的一个生成元，签名者的私钥为 x 满足 $1<x<p-1$，签名者的公钥为 $y=g^x \bmod p$。

明文信息为 M，计算 $H(M)$。随机选择整数 k，k 与 $p-1$ 互质，计算

$$r = g^k \bmod p$$

求解

$$s = (H(M) - xr)k^{-1} \bmod (p-1)$$

将明文信息 M 和数字签名(r,s)发送给验证者。

验证者接收到 M 和数字签名(r,s)后验证等式

$$y^r r^s \bmod p = g^{H(M)} \bmod p$$

若成立，则接受签名；否则拒绝签名。

3.4.3 数字签名算法 DSA

DSS(Digital Signature Standard，数字签名标准)由美国国家标准技术研究所(NIST)于

1991 年 8 月颁布，该标准采用的数字签名算法为 DSA(Digital Signature Algorithm)。

DSA 签名算法的参数定义和执行过程：

- 随机选取素数 p，满足 $2^{l-1} \leqslant p \leqslant 2^l$，$l$ 是 64 的倍数且 $512 \leqslant l \leqslant 1024$。
- q 为素数，能够整除 $p-1$，且 $2^{159} \leqslant q \leqslant 2^{160}$。
- $g = h^{\frac{(p-1)}{q}} \bmod p$，其中 h 满足 $1 < h < p-1$，且 $g > 1$ 的任意整数。
- 随机选择 x，满足 $0 < x < q$。
- $y = g^x \bmod p$，其中 $\{p, q, g, y\}$ 为签名者公钥，而 x 为签名者私钥。
- 随机选择 k，满足 $0 < k < q$。

DSA 中信息 M 的数字签名为两个 160 比特的证书 r 和 s，$H(x)$ 为数字摘要算法 SHA-1，数字签名如下

$$r = (g^k \bmod p) \bmod q$$
$$s = (k^{-1}(H(M) + xr)) \bmod q$$

验证者收到数字签名 $\{M, s, r\}$，并验证

$$w = s^{-1} \bmod q$$
$$u_1 = (H(M)w) \bmod q$$
$$u_2 = (rw) \bmod q$$
$$v = ((g^{u_1} y^{u_2}) \bmod p) \bmod q$$

如果 $v = r$，那么数字签名验证有效；否则说明与原信息 M 不一致，或者数字签名是伪造的。

3.4.4 特殊用途的数字签名算法

1. 盲签名

盲签名最初是由 David Chaum 于 1982 年提出的，其主要特征是在隐藏消息内容的情况下使签名者对消息实施数字签名，即当 A 方要求 B 方给指定信息签名时，不需要将信息内容泄露给 B 方。盲签名的出现满足了签名所有者的匿名要求，例如匿名的电子投票系统中，系统既要保证投票的合法性、准确性，又要保证投票者的匿名性。盲签名广泛地应用于电子货币，消费者在提取电子货币时需要有银行的数字签名，但是对银行隐藏电子货币提取人的身份，以实现电子交易的不可跟踪性。

因此，除满足普通签名的条件之外，盲签名还必须满足以下两个条件：

(1) 签名者无法看到其签署消息的具体内容。

(2) 即使公布数字签名信息，签名者也无法跟踪数字签名相关信息。

David Chaum 基于 RSA 算法设计实现盲签名，假设 A 需要 B 对信息 M 实施数字签名，其中 B 的密钥对为 (pk, sk)，签名系统的模数为 n，那么盲签名的过程如下：

(1) A 随机选择整数 r，满足 $\gcd(r, n) = 1$。

(2) A 计算 $x = r^{pk} M \bmod n$，将 x 发送给 B，而 B 无法根据 x 计算 r。

(3) B 计算 x^{sk}，并发送给 A。

(4) A 计算 $r^{-1} x^{sk} = r^{-1}(r^{pk} M)^{sk} = r^{-1} r^{pk \cdot sk} M^{sk} = M^{sk}$，获得 B 对 M 的签名 M^{sk}。

盲签名通过对签名者 B 隐藏部分被签名信息的内容实现签名的匿名性，并使签名者难

以伪造隐藏部分的信息。盲签名的基本流程如下：

(1) A 将明文信息 M 乘以一个随机数 t(盲因子)，形成一个新信息 M' 并发送给 B。

(2) B 对信息 M' 实施签名，得到 sig(M')发给 A，但 B 不能获得 M 的内容。

(3) A 去除签名 sig(M')中的盲因子，得到 B 对明文信息 M 的数字签名 sig(M)。

2. 代理签名

代理签名也称为委托签名，是指签名者因为主观或客观原因无法行使签名权，而将签名权授权给代理人，委托其替代自己行使签名权。代理签名的主要特点如下：

- 不可伪造性。只有原签名人和代理人能够实施合法的代理签名。
- 可区分性。本人签名和代理签名是可以区分的。
- 可识别性。原签名人可以通过代理签名信息识别出代理人的身份。
- 不可抵赖性。代理人无法事后抵赖已做出的代理签名。
- 可证实性。验证者能够证实原签名人认同了代理人的签名。

代理签名按照密钥对管理方式分为完全委托和部分委托。

- 完全委托是指原签名人将自己的签名密钥对直接交给代理人，则意味着签名权的全部移交。完全委托方式要求原签名者对代理人签名的完全信任，完全委托签名方式如图 3-29 所示。

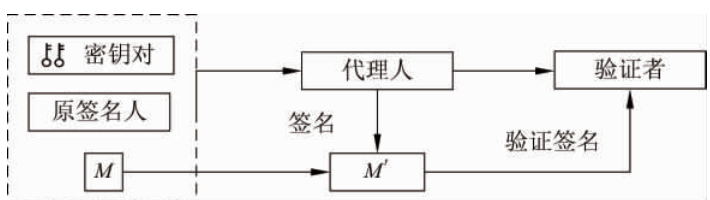

图 3-29 完全委托方式

- 部分委托方式是指原签名人和代理人分别拥有自己的密钥对，可以明确区分责任，并且原签名人可以通过代理签名信息识别出代理人的身份，部分委托签名方式如图 3-30 所示。

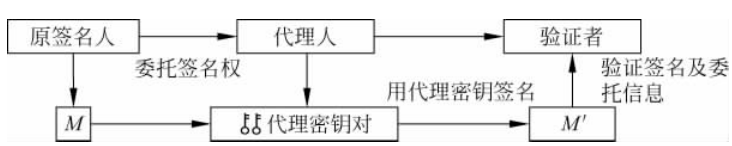

图 3-30 部分委托方式

由 K. Zhang 提出的基于离散对数问题的代理签名方案如下：

随机选取素数 p 和 q，且满足 $q \mid p-1$，整数 g 是 Z_p^* 的一个 q 阶生成元。A 随机的选择 $x \in Z_q^*$，并计算 $y=g^x \bmod p$，x 和 y 分别为 A 的私钥和公钥。签名人 A 将签名权授权给代理人 B 的过程如下：

- A 随机选取 $k' \in Z_q^*$，计算 $r'=g^{k'} \bmod p$，并将 r' 发送给 B。
- B 随机选取 $\alpha \in Z_q^*$，计算 $r=g^{\alpha} r' \bmod p$，并将 r 发送给 A。

- A 计算 $s'=rx+k'\bmod q$，并将 s' 发送给 B。
- B 计算 $s=s'+\alpha\bmod q$。
- 若授权签名(r,s)使等式 $g^s\bmod p=y^r r\bmod p$ 成立，则将 s 作为代理人 B 的私钥，将 $y_B=g^s\bmod p$ 作为对应的公钥。

3. 群签名

群签名也称为团体签名，是由 Chaum 和 Van Heyst 于 1991 年提出的，是指群中的任意成员可以以匿名的方式代表整个群对明文信息进行数字签名。验证者可以使用群成员共有的群公钥对数字签名进行认证。群签名具有匿名可撤销性，通常群签名的接收者能验证群签名的有效性，但是不能确定群签名成员的身份，但是当群签名出现争议时，可以恢复出签名者的身份。

群签名方案广泛地应用于需要多方签名的安全通信系统中，例如多银行的电子现金系统，可以利用群盲签名来构造由多银行参与发行的电子货币，银行间可以相互认证电子现金，能够实现电子现金的跨行结算，并且在中央银行的统一控制下发放和使用电子现金。

3.4.5 椭圆曲线数字签名算法

ECC(Elliptic Curve Cryptography，椭圆曲线加密)属于公钥加密体制，与 RSA 等加密体制相比具有加密密钥短、安全强度高的特点。ECDSA(Elliptic Curve Digital Signature Algorithm，椭圆曲线数字签名算法)和 RSA、DSA 的功能基本相同，并且基于椭圆曲线的数字签名和验证的效率远高于基于 RSA 和 DSA。下面简单介绍 ECDSA 的参数和算法结构。

1. ECDSA 的全局参数

ECDSA 的域参数由一条特征为 p 的有限域 F_q 上的椭圆曲线 E 和基点 $G\in E(F_q)$ 组成。

(1) 有限域大小为 q。基于 F_p 存在 $q=p$，其中 q 为奇素数。若基于 F_{2^m}，则 $q=2^m$。

(2) 用 FR 来描述 F_q 中的一个元素。

(3) 比特串 seedE 的长度至少为 160 比特。

(4) $x_G,y_G\in F_q$，$G=(x_G,y_G)$，$G\in E(F_q)$，G 的阶为素数 n，$n>2^{160}$，且 $n>4\sqrt{q}$。

(5) $a,b\in F_q$，定义椭圆曲线方程 $y^2=x^3+ax+b$ 或 $y^2+xy=x^3+ax^2+b$。

(6) $h=\dfrac{\#E(F_q)}{n}$。

上述参数可以表示为 $D=(q,FR,a,b,G,n,h)$。

2. 产生公钥和私钥参数

ECDSA 的公钥是椭圆曲线上的一个点，私钥是小于椭圆曲线阶 n 的整数。对于每个私钥和公钥对，都存在一个对应的全局参数组(q,FR,a,b,G,n,h)。密钥对的产生过程如下：

(1) 选择一个随机数 d，$d\in[1,n-1]$。

(2) 计算 Q，$Q=dG$。那么公钥为 Q，私钥为 d。

3. ECDSA 的算法描述

用 U 表示签名方，用 V 表示验证方，数字签名初始化过程如下：

(1) U 建立椭圆曲线域参数 $T=(p,a,b,G,n,h)$。

(2) U 建立自己的密钥对(d_u,Q_u)，$Q_u=d_uG$。

(3) U 选择一个 Hash 函数，并通过可靠的方式将所选择的 Hash 函数和建立的椭圆曲线域参数 T 传递给 V。

签名方 U 数字签名过程如下：

(1) 选择临时密钥对(k,R)，其中 $R=kG=(x_R,y_R)$和域参数 T 相关。

(2) 令 $r=x_R \bmod n$，如果 $r=0$，返回上一步。

(3) 计算明文信息的 Hash 值 $e=H(M)$。

(4) 计算 $s=k^{-1}(e+rd_u)\bmod n$，如果 $s=0$，返回到步骤(1)。

(5) 输出 $S=(r,s)$为数字签名。

验证方 V 验证从签名方 U 发来的数字签名是否正确，从而判断接收到的消息是否真实，或者对方是否为真实的实体。验证过程如下：

(1) 如果 $r,s\notin[1,n-1]$，签名信息验证错误。

(2) 计算明文信息的 Hash 值 $e=H(M)$。

(3) 计算 $u_1=es^{-1}\bmod n$，$u_2=rs^{-1}\bmod n$。

(4) 计算 $R=(x_R,y_R)=u_1G+u_2Q_u$。

(5) 计算 $v=x_R \bmod n$，如果 $v=r$，验证成功，否则验证失败。

3.5 密钥管理技术

密钥管理是对信息安全加密体制所产生的密钥采取安全有效的管理措施，保证密钥在应用和传输过程中的安全性，主要包括密钥的生成与存储、密钥的分配、密钥的更新与撤销、密钥共享、会议密钥分发以及密钥托管等，其中公钥加密体制是制定密钥管理标准和规范的基础。

3.5.1 密钥管理技术基础

密钥管理是整个加密、解密系统中最重要的也是最易受到攻击的环节，攻击者直接入侵管理系统获得密钥远比破译加密算法所花费的代价小得多，因此攻击者往往选择窃取系统密钥，密钥的泄露将会直接导致整个安全通信系统失效。

密钥管理涵盖了密钥的整个生命周期，涉及密钥的生成、分发、存储、验证、传输以及销毁，密钥管理系统的主要目的是保护密钥安全和提供正常的密钥服务。密钥管理技术主要分为对称密钥管理技术和公开密钥管理技术两大类。

1. 对称密钥管理技术

由于对称加密体制的首次密钥传输问题，通常对称加密体制的通信双方利用公钥加密

技术来传输对称加密的密钥。因此在实际通信系统中通信双方充分利用公钥加密算法和对称加密算法的优势实现安全通信，即充分利用对称加密算法的高效性和公钥加密算法的身份可认证性。

2. 公开密钥管理技术

公开密钥管理技术主要是应用数字证书管理体系来实现密钥的安全管理，可以实现密钥的生成、分发、存储、验证、传输以及销毁等。公开密钥管理技术主要是利用公钥加密算法的保密机制和签名与认证机制来实现密钥的安全管理。

3.5.2 RSA 密钥管理技术

RSA 密钥管理技术主要是利用 RSA 加密算法来实现通信密钥安全传输与管理。通信双方的密钥对由 KDC(Key Distribution Center，密钥分发中心)产生和分发，其中 RSA 加密算法的公钥为 $PK=\{e,n\}$ 和私钥为 $SK=\{d,n\}$。每次进行安全通信时，发送方随机产生一个大数作为会话密钥 S，并依次使用自己的私钥和接收方的公钥对会话密钥 S 进行加密，然后传给接收方。收到信息后，接收方依次使用自己的私钥和发送方的公钥进行解密，得到会话密钥 S。通信双方在协商好会话密钥 S 后，使用会话密钥 S 进行保密通信。这种方法既保证了会话密钥 S 的安全性，又能实现每次通信都能够使用不同的密钥。

3.5.3 Diffie-Hellman 密钥交换协议

Diffie 和 Hellman 在 1976 年提出一个安全的密钥交换算法，密钥交换算法的安全性基于求解有限域上离散对数的困难性。Diffie-Hellman 密钥交换算法如下：

(1) A 表示发送方，B 表示接收方，A 和 B 随机选择大素数 q 和 $GF(q)$ 上的本原元 g，其中 $1<g<q$，并公开 q 和 g。

(2) 发送方 A 随机选择 $X_A\in Z_q^*$，计算并公开 $Y_A=g^{X_A}\bmod q$。

(3) 接收方 B 随机选择 $X_B\in Z_q^*$，计算并公开 $Y_B=g^{X_B}\bmod q$。

(4) A 计算产生密钥 $K=(Y_B)^{X_A}\bmod q$，B 计算产生密钥 $K=(Y_A)^{X_B}\bmod q$。那么密钥 K 就是共享的密钥。A 的密钥 $K=(Y_B)^{X_A}\bmod q$ 等于 B 的密钥 $K=(Y_A)^{X_B}\bmod q$。

证明：

$$\begin{aligned}K&=(Y_B)^{X_A}\bmod q\\&=(g^{X_B}\bmod q)^{X_A}\bmod q\\&=(g^{X_B})^{X_A}\bmod q\\&=g^{X_BX_A}\bmod q\\&=(g^{X_A}\bmod q)^{X_B}\bmod q\\&=(Y_A)^{X_B}\bmod q\end{aligned}$$

3.5.4 PGP 密钥管理

PGP(Pretty Good Privacy)是利用公钥加密算法、对称加密算法和数字摘要算法来实现网络安全通信的软件系统，通常应用 RSA 公钥加密算法、AES 对称加密算法和 SHA 数

字摘要算法实现保密通信、数字签名和密钥管理等功能，主要保护对象包括电子邮件、文件、磁盘以及网络通信数据的安全。

PGP 使用 RSA 和 DSA 数字签名算法对数据实施数字签名，使用 AES、IDEA 对称加密算法对通信数据加密，使用 SHA 数字摘要算法计算通信数据的数字摘要值。为了提高数据处理速度，PGP 在传输数据之前，首先使用数据压缩技术对通信数据进行压缩，压缩率通常为 50%，这样既提高了加密速度，又能节约大量的网络带宽。

PGP 软件可以加密电子邮件，同时邮件接收者可以通过验证邮件的数字签名来确定邮件发送方的真实身份。此外，PGP 软件的一个重要应用是磁盘加密，主要包括本地磁盘加密、网络共享磁盘加密和虚拟磁盘加密。PGP 可以对本地的整块磁盘进行加密，要访问磁盘数据必须提供正确的密码，同时 PGP 可以像加密本地磁盘一样对网络共享的磁盘实施加密。虚拟磁盘加密是将磁盘部分空间虚拟为一个加密磁盘驱动器，并添加文档和文件夹到虚拟磁盘驱动器中，PGP 会自动处理虚拟磁盘驱动器中文件的加密和解密。

3.6 数字认证技术

数字认证技术是实现网络安全通信的基础，公共网络中通信数据的完整性校验和通信实体的身份认证是验证通信数据的正确性和来源的真实性的具体技术措施。数字认证技术主要包括消息认证技术、身份认证技术以及身份认证协议等。

3.6.1 消息认证

消息认证是对通信数据的真实性和完整性进行验证，主要目的是使数据的接收者能够验证其所接收的信息是否可信。消息认证包括两方面内容：一方面是验证消息的内容是否真实；另一方面是验证消息源是否可靠。此外，有时也会验证消息的时效性和顺序。

通常加密算法均具有消息认证功能，因为只有合法用户才能解密并获得正确信息。另外，若消息中包括校验码或者时间戳，就能准确地判断消息在传输过程中是否被篡改或者消息是否能够按预期到达。

1. 消息认证码

发送方 A 和接收方 B 若要传输消息 M，首先要协商会话密钥 K_{AB}，然后 A 计算消息验证码 $MAC_M=F(K_{AB},M)$，并且将其附加在消息 M 后一起传送给接收者 B。B 计算消息验证码 $MAC_M=F(K_{AB},M)$，并且将计算得到的 MAC_M 和接收到的 MAC_M 进行比较，如果相同说明消息完整，即接收者能够确认消息没有被篡改、消息来自合法的发送者和消息序列正确，具体过程如图 3-31 所示。

2. 单向散列函数

相对于消息认证码，单向散列函数具有更高的效率，并且应用单向散列函数计算散列值时不需要密钥。单向散列函数已经成为消息认证码的替代算法，主要用于通信数据的完整性校验。通常单向散列函数与对称加密算法和公钥加密算法相互结合使用，主要应用形式如下：

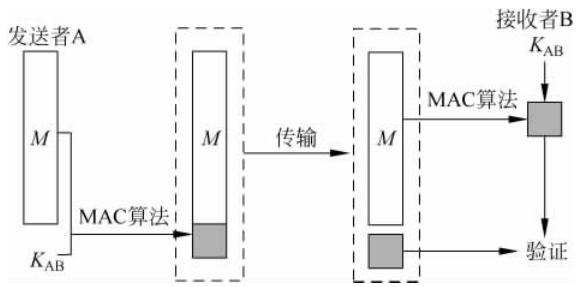

图 3-31 消息认证过程

第一种方式是单向散列函数与对称加密算法结合使用，发送者首先计算明文信息的散列值，然后用对称加密算法加密散列值，并且与明文信息一起发送给接收方，认证过程如图 3-32 所示。

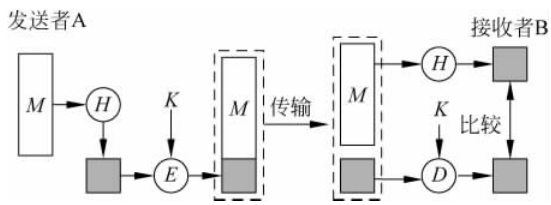

图 3-32 散列函数与对称加密算法结合

第二种方式是单向散列函数与公钥加密算法相结合实现通信数据的完整性校验，发送者首先计算明文信息的散列值，然后用接收方的公钥对散列值进行加密，并将其附在明文信息后发送给接收者，基本过程如图 3-33 所示。

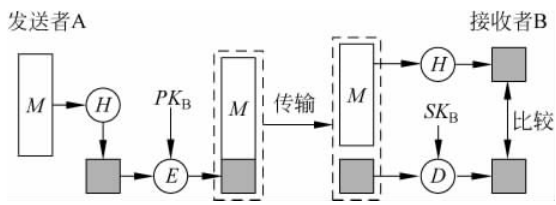

图 3-33 散列函数与公钥加密算法结合

第三种方式是单向散列函数与共享密钥相结合，发送方将共享秘密 S_{AB} 与明文信息合并，并使用单向散列函数计算散列值 $MD_M=H(S_{AB}||M)$，并将其附在明文信息后发送给接收者，基本过程如图 3-34 所示。

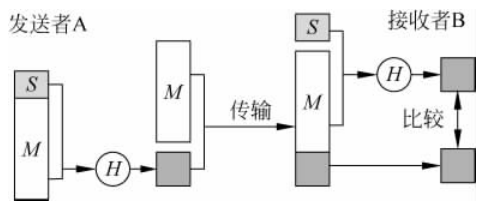

图 3-34 单向散列函数与共享密钥结合

3.6.2 身份验证技术

身份验证是指网络通信系统的实体身份认证，主要包括用户、服务器和其他网络终端设备。身份认证的主要目的是验证实体身份的真实性和合法性，认证对象主要包括用户密码和口令、身份证、护照、数字证书以及用户的基本特征（指纹、视网膜、DNA、声纹等）。

身份认证协议的主要形式有：

（1）双向认证协议——通信双方相互认证彼此身份的真实性和合法性。

（2）单向认证协议——仅通信的一方认证另一方的身份，例如服务器验证用户身份的合法性，但不对用户提供服务器身份认证。

（3）基于对称密钥的认证协议——利用对称加密体制实现网络通信实体身份认证，通信双方需要事先约定好共享密钥。

（4）基于公钥的认证协议——利用公钥加密体制实现网络通信实体身份认证，通常通信双方利用对方公钥对实体身份进行认证。

3.6.3 Kerberos 身份验证协议

Kerberos 协议是由麻省理工学院提出的网络通信实体身份验证协议，其主要目标是为网络客户机和服务器提供身份认证服务，主要特点是用户只需输入一次身份验证信息就可以访问系统中多个服务。

1. Kerberos 协议

Kerberos 协议主要应用在分布式的客户/服务器体系结构中的实体身份认证，系统可以采用一个或多个 Kerberos 服务器提供身份鉴别服务。

首先客户机向 Kerberos 认证服务器提供身份验证信息，通过 Kerberos 服务器认证后，客户机获得用于身份证明的电子票据，此后，客户机可以凭借此票据访问系统的多个服务器，而无须在登录每个服务器时都输入一次身份验证信息。Kerberos 协议通过集中管理用户身份认证，提高了身份认证效率和安全性，可以有效地防止客户机重复利用票据登录服务器，并且可以防止非法用户伪造票据。Kerberos 协议的基本结构如图 3-35 所示。

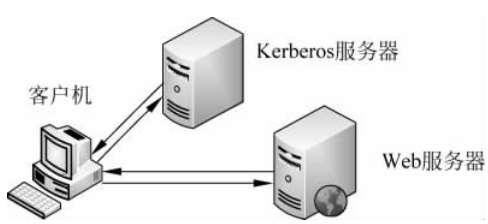

图 3-35 Kerberos 协议的基本结构

2. Kerberos 协议的术语

通常描述 Kerberos 协议需要掌握的术语如下：

主体——参与网络通信的实体，是具有唯一标识的客户机或服务器。

认证——验证通信实体所宣称身份的真实性和合法性。

认证头——身份认证的票据和提交给服务器的认证码。

认证路径——实现跨域身份认证时，所经历中间域的路径信息。

认证码——网络通信实体身份认证信息，内容主要包括时间戳、客户机地址和会话密钥等。

票据——Kerberos 协议中客户机用于证明其真实身份的电子证据，主要包括客户机身份标识、会话密钥、时间戳和其他信息。

会话密钥——任意两个通信实体之间实现相互通信而协商的临时对称加密密钥，只使用一次，使用结束立即作废。

密钥分发中心 KDC——Kerberos 协议中负责分发票据和会话密钥的可信网络服务中心。

Kerberos 协议主要包括认证服务器、票据授予服务和应用服务器，其中认证服务器和票据授予服务器为通信实体提供身份认证服务，应用服务器为用户提供最终的应用服务。

3.6.4 基于生物识别技术的身份验证技术

基于生物识别技术的身份验证是利用人体固有的生理特征和行为特征的统计特性对通信实体的身份实施认证的技术，生物特征主要包括指纹、脸型、虹膜、笔迹和语音等，利用计算机与光学、声学、生物传感器将生物统计学特征转换为电子信号进行身份信息的存储和鉴别。基于生物特征的识别系统将采集的个体特征和事先存储在认证服务器的特征进行比对来验证实体身份的真实性和合法性。

基于生物特征的身份识别技术简化了密钥管理和分发机制，将生物特征作为身份识别的依据，可对网络通信数据实施数字签名和认证。基于生物特征的身份识别系统的工作原理如图 3-36 所示。

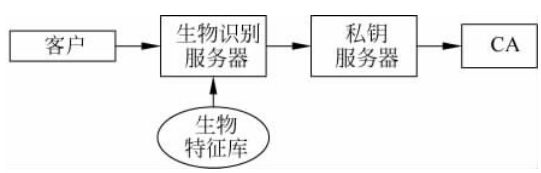

图 3-36 基于生物特征的身份识别系统

基于生物特征的身份识别系统的工作原理是：首先利用生物特征读取终端采集生物特征信息，并将其传送到认证服务器进行认证。认证服务从生物特征库中提取特征与采集到的用户特征进行比对，并判断用户特征是否有效，然后签署认证信息给私钥服务器，私钥服务器计算文档信息的数字摘要并向用户返回一个数字签名，这样就在用户与服务器间建立了完全信任的关系。基于生物特征的身份识别技术能够有效降低密钥管理的复杂度和成本，并且能够应用于各种网络服务系统的实体身份认证。

本章小结

本章主要介绍了信息安全技术基础知识，主要包括信息安全通信的基本原理、密码学技术基础、数字签名技术、密钥管理技术以及数字认证技术等。重点掌握对称加密算法、公钥加密算法和数字摘要算法的基本结构和原理；熟练掌握数字签名技术的基本原理和常用数字签名算法；熟练掌握密钥管理技术在维护网络系统安全中的重要作用；掌握数字认证的基本原理，了解常用的消息认证方法和基于生物特征的身份认证方式。

本章习题

1. 请简述对称加密算法和公钥加密算法的基本特征，并说明它们各自的优点是什么。
2. 请简述分组密码的四个工作模式，并说明它们各自的特点。
3. 请简述 DES 加密算法和 RSA 加密算法的基本流程。
4. 请简述安全散列算法 SHA 的基本流程。
5. 请简述 Diffie-Hellman 密钥交换协议的基本流程。
6. 请简述基于椭圆曲线加密体制的数字签名算法的基本流程。